Scibox: Online Sharing of Scientific Data via the Cloud

Jian Huang*
jian.huang@gatech.edu

Xuechen Zhang*
xczhang@cc.gatech.edu

Greg Eisenhauer*
eisen@cc.gatech.edu

Karsten Schwan*
schwan@cc.gatech.edu

Matthew Wolf*§
mwolf@cc.gatech.edu

Stephane Ethier†
ethier@pppl.gov

Scott Klasky§
klasky@ornl.gov

*Georgia Institute of Technology , Atlanta, USA
†Princeton Plasma Physics Laboratory , Princeton, USA
§Oak Ridge National Laboratory , Oak Ridge, USA

ABSTRACT

Collaborative science demands global sharing of scientific data. But it cannot leverage universally accessible cloud-based infrastructures like DropBox, as those offer limited interfaces and inadequate levels of access bandwidth. We present the Scibox cloud facility for online sharing scientific data. It uses standard cloud storage solutions, but offers a usage model in which high end codes can write/read data to/from the cloud via the APIs they already use for their I/O actions. With Scibox, data upload/download volumes are controlled via D(ata)R(eduction)-functions stated by end users and applied at the data source, before data is moved, with further gains in efficiency obtained by combining DR-functions to move exactly what is needed by current data consumers. We evaluate Scibox with science applications and their representative data analytics – the GTS fusion – demonstrating the potential for ubiquitous data access with substantial reductions in network traffic.

1. INTRODUCTION

Global, distributed scientific processes critically rely on the data generated by scientific simulations and instruments. Examples range from investigations by science teams to a plethora of enterprise applications. Common to all such endeavors is the need for convenient and ubiquitous data sharing. The goal of this work Scibox is to leverage the ease of use and universal accessibility of commercially developed cloud data sharing software, like DropBox to provide scalable scientific data sharing infrastructure. Scibox (1) proposes a science data usage model and (2) offers methods to circumvent unnecessarily large data uploads to or downloads from the cloud. Concerning (1), Scibox presents to data producers and consumers the standard I/O APIs already used by science applications, like the ADIOS I/O infrastructure. As a result, science codes can write output to the cloud that can then be directly read by subsequent, potentially remote data analytics or visualization codes, in the same fashion as I/O and subsequent analysis are being performed in today's high end facilities used for running science simulations. Concerning (2), to reduce cloud data upload/download volumes, Scibox permits an end user to identify the exact data needed for each specific inquiry, by specifying the D(ata)R(eduction)-function that is applied at the data source and before data is actually uploaded to the cloud. In addition and for efficient online data sharing across multiple concurrent science end users, Scibox then combines users' different DR-functions into a cumulative data reduction method, in order to upload to the cloud only those data items needed by the complete current set of data sharing clients. We conduct extensive evaluation using both synthetic and real scientific workloads, including GTS physics diagnosis and combustion image processing. Our results show that Scibox can reduce networking traffic by up to 65X. In addition, Scibox users can have a much better cloud experience with less I/O latency and significantly saved usage cost.

2. Scibox SYSTEM OVERVIEW

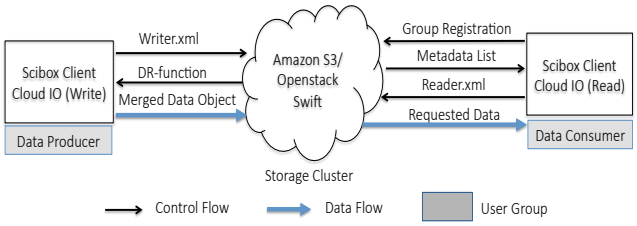

Figure 1: Scibox architecture

Key to Scibox is the simple question of: *how to design a scientific data sharing infrastructure so as to reduce data transfer and storage sizes?*, and the approach taken to answer this question is one that *stores only the data desired by end users in the cloud.* The Scibox realization of this solution approach is shown in Figure 1, along with its control and data flows. The current implementation of Scibox uses the Swift object store to implement data storage on some **storage cluster**. This means that the files produced by ADIOS write actions are stored as objects that can be accessed by users through unique URLs. The **data producers** are scientific simulations or instruments, which repeatedly and/or periodically produce output data to which users apply various analytics actions/codes. We assume such data to be locally stored and available for some time, and we assume the presence of producer-side compute capacity for data cleaning, preconditioning, and for the analytics needed to select/filter data, thus preparing it for the broader degree of data sharing enabled by Scibox. **Data consumers** download desired data from the cloud, where a **user group** is composed of some single data producer and multiple consumers who share select content. Multiple groups can be created for sharing different data sets produced by an application or instrument. For all data sharing, however, Scibox client functions are running on both the producer and consumer sides, the latter enabling the data reductions desired by end users. The Scibox control flows, therefore, span producers and consumers.

The current Scibox prototype supports DR-functions classified

into eight categories. Functions of types $DR1$ to $DR4$ – *basic DR-functions* – describe user requirements regarding a single variable. Functions of types $DR5$ to $DR7$ describe more complex relationships between variables. For example, a user of GTS data can take advantage of $DR7$ when the ion's temperature data is needed only when its velocity is larger than some threshold. Finally, more advanced users can explicitly define their own, custom DR–functions – Type $DR8$ – using the Co(n)D(emand) [1] programming language. CoD is used because its simple code generation facilities can be run at consumer, producers, or 'in' the cloud, across the entire set of participants in a Scibox system. The system operates by registering a string describing the DR-function at data producers, then compiling and running the function 'on demand' at the producer, on the specified input data. For such custom functions, Scibox does not guarantee them to be executable for arbitrary input data, e.g., if there are mismatches in the function's assumptions concerning input data types and sizes with the actual data seen, function execution will fail, returning the original data to the user.

Scibox is implemented with the OpenStack Swift object store [2] software and the ADIOS [3] library. The ADIOS library supports multiple I/O transports, e.g., MPI and POSIX, for users with different I/O requirements. Such transport methods are configured in an XML file, so that a change in transport does not require developers to modify or recompile their applications. Scibox contributes to ADIOS a new Cloud-IO transport so that I/O requests can be sent to the cloud with the same APIs already used by end users for constructing their I/O pipelines on high end machines. The transport is implemented so that application-level data buffers can be directly used as inputs for Scibox's DR-functions.

3. PERFORMANCE EVALUATION WITH GTS

The performance of Scibox is evaluated in a private cloud comprised of multiple clusters located across multiple campuses. Jedi runs the Swift store, acting as the cloud service provider, with three nodes and a total disk capacity of up to 2.4 TB. The Vogue cluster, which has 10 eight core servers, is used to run applications that produce scientific data using the ADIOS library. To simulate data access with different network latencies via the Internet, we run two remote Scibox consumer clients at Wayne State University (WSU), , and at The Ohio State University (OSU), together with local clients run at Georgia Tech (GT). We use the latest stable version of OpenStack Swift.

In this experiment, we focus on sharing checkpointing file restart.bp generated by the GTS petascale application [4]. The file includes all the variables necessary to restart a simulation, the most important of which are the $zion$ and phi arrays. Analytics are performed by Scibox consumers to calculate the particles' spatial distribution based on the $zion$ arrays, and to execute Fourier transforms or histograms for physics diagnosis on phi. By default, the two arrays are stored together along with many other variables in the file restart.bp. With Scibox, the consumers can filter out the data which are not needed by the analytics using DR-functions, e.g., **I (zion, *DR*1, MAX)** and **II (phi, *DR*8, Histogram)**. We run the GTS application on the Vogue cluster with 128 parallel processes and 5 particles per grid cell. The total amount of raw data that are generated by GTS for software restart is 908 MB for the chosen problem size. We set up 30 Scibox consumers to download the data from three locations. Every consumer requested both $zion$ and phi variables using either DR-function I or II. At each location, there are ten consumers. Because of the firewall issue, data cannot be transferred from Jedi to their remote clients using curl, instead FTP is used for data transfer. For Scibox the related networking latency is calculated based on the reduced data size and its measured networking bandwidths, which are 900 KB/s, 4.4 MB/s and 44 MB/s from the Jedi object store to WSU, OSU, and GT, respectively. We assume that the data are directly pushed to a user's local storage, as soon as it is generated. Therefore, user experienced latency includes the GTS computation time, the DR-function execution time for Scibox users, data downloading time, and post-computation time for non-Scibox users. Results are shown in Figure 2.

Following observations may be made. First, Scibox consistently achieves better performance than FTP-based, direct transfer solutions, because only the results of DR-functions are transferred via the cloud. Another observation is that although network bandwidth is the bottleneck, e.g., for WSU users, Scibox still achieves consistently short latency. This is because for these use cases, due to the relatively small amounts of data transferred, Scibox user experienced latencies are primarily determined by GTS and DR-function execution times. As with our our experience with other DR-functions, it is clear that for remote users, filtered data sizes are a significant determinant of the performance benefits obtained from using Scibox.

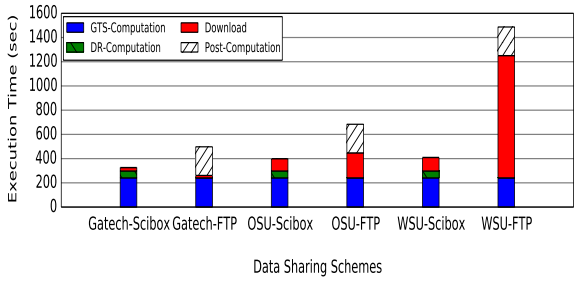

Figure 2: Comparison of user experienced latency in the GTS experiment.

4. CONCLUSIONS

Scibox is a scientific data sharing infrastructure able to operate across both public or private cloud stores. New functionality offered by Scibox exploits the structured nature of science data by sharing metadata across end users before actual data is moved and by then permitting end users to control the amounts of data moved. Scibox is implemented with the ADIOS I/O library, permitting current ADIOS users to seamlessly extend their I/O pipelines across cloud-based infrastructures. Experimental results demonstrate improved end-to-end latencies for data moved from sources to sinks, across the cloud, compared to using the methods now employed by commercial systems like Dropbox.

5. REFERENCES

[1] G. Eisenhauer, M. Wolf, H. Abbasi, and K. Schwan, "Event-based systems: Opportunities and challenges at exascale," in *DEBS'09: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, 2009.

[2] OpenStack LLC, "Openstack install and deploy manual," 2012.

[3] N. Podhorszki, Q. Liu, J.Logan, H. Abbasi, J. Y. Choi, and S. Klasky, "Adios," *ADIOS 1.4.0 Developer's Manual*, 2012.

[4] K. Madduri, K. Ibrahim, S. Williams, E. IM, S. Ethier, J. Shalf, and L. Oliker, "Gyrokinetic toroidal simulations on leading multi- and manycore hpc systems," in *ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2011.