# ParSA: High-throughput Scientific-data Analysis Framework with Distributed File System

Tao Zhang

*Tsinghua*

## Abstract

Scientific-data analysis and visualization has become a key component in nowadays large-scale simulations. Due to the rapidly increasing data volume and awkward I/O pattern among high-structured files, known serial methods/tools cannot scale well and usually lead to poor performance over traditional architectures. In this paper, we propose a new framework: ParSA (parallel scientific-data analysis) for high-throughput and scalable scientific analysis, with distributed file system. ParSA present the optimization strategies oriented for physical disk to maximize distributed I/O property of distributed file system as well as to maximize overlapping the data reading, processing and transferring during computation. Besides, ParSA provides the similar interfaces as the NCO (netCDF Operator), which is used in most of climate data diagnostic package, making it easy to port this framework. We utilize ParSA to accelerate well-known analysis methods for climate models on Hadoop Distributed File System (HDFS). Experimental results demonstrate the high efficiency and scalability of ParSA.

*Keywords:*
Data intensive, Scientific data analysis, Distributed file system

## 1. Introduction

In most of modern scitific applications, huge amounts of data are produced. Large-scale simulations, such as climate modeling, high-energy physics simulation and genome mapping, generate hundreds of terabytes data volumes(Tevfik, 2009; Hey, 2003). Additionally, it still increases as the high resolution model developping. As a consequence, analysis of scientific-data is data-intensive.

In fact, almost all of scientific-data are stored in high-structured files, some of which provide parallel I/O interface, such as Network Common Data Format version 4(NetCDF4), Hierarchical Data Format 5(HDF5) (hdfs) and ADIOS BP data format (bp) (bp), and some of which only support serial I/O interface, like Network Common Data Format version 3(NetCDF3). All of these are self-describing, machine-independent data format.

In scientific-data analysis, large-scale scientific-data are stored in RAID-5/6 or parallel file system. Yet the analysis methods/ tools are always centralized approaches, such as NCO and NCL, which are the most used in climate applications for prossing NetCDF files, leading to very poor scalability and time-consuming performance.

Inspired by big data solution in Internet Port Data Center (IPDC), numerous frameworks with distributed strategy have been developed. MapReduce is a program framework for processing and generating large data sets, provideing automatic parallel mechanism and build-in fault-tolerance on a cluster. However, this solution with MapReduce requires that data first be transformed into a text-based format. SciHadoop is a Hadoop plugin allowing scientists to specify logical queries over array-based data models. It executes queries as map/reduce programs defined over the logical data model. It shows remarkable improvements for holistic functions of NetCDF data sets for the following optimization goals: reduce total data transfers, reduce remote reads and reduce unnecessary reads. Nevertheless, SciHadoop using java language leads to the compatibility problem to the existing climate data analysis tools, which is written by c shell scripts, NCL and NCO commands. The SWAMP project [9] has provided the parallel NCO operations, but the reading performance is still bottleneck.

In this paper, we propose a new framework - Parallel Scientific-data Analysis (ParSA). We utilize the distributed I/O property with Hadoop Distributed File System (HDFS) to improved data reading throughput. What is more, ParSA optimizes the data layout schedule stored in the distributed file system to overlap the data reading, processing and transferring. Besides, it provides parallel NCO operations, cooperating with HDFS, making it easy to use the efficient tool, without changing a lot for current climate analysis package.

## 2. HDFS and scientific-data analysis

In this section, we will present the property of distributed file system HDFS, relicas and scheduler, which can be taken advantage of to optimize distributed I/O performance. We also present the character of scientific-data analysis, and discuss about the probability of analysis transportaion onto HDFS.

### 2.1. HDFS

HDFS is an open source project, driven by Google File System (GFS). As a distributed, scalable and portable file system, HDFS is inherent for large-scale data-intensive process.

In HDFS, there are two types of node, Namenode and Datanode. Namenode maintains file system tree and metadata for all files or directories stored in HDFS, and Datanodes are where the data are actually stored. When a file are stored into HDFS, it will be split into file blocks as the storage unit of HDFS. For achieving fault-tolerance, HDFS stores three replicas, by default, for each file blocks in different datanodes. Therefore, even individual node halts down, all data, which are stored in the halted node, can be accessed from other replicas. All of I/O operates can be manipulated through Clientnode.

Each Datanode can mount several hard disks and it manages these hard disks by itself. By default, Datanode will store each block into the hard disks in a round-robin way. For example, when a file "a" will be stored into HDFS via Clientnode, Namenode will add "a" into the file system tree. Then Clientnode begin writing the content into HDFS. Once Clientnode detects that current writing size exceeds the block size, 64MB by default, it will ask Namenode for a new block with unique block number. Simultaneously, Namenode need recode mapping relation between "a" and block numbers. Since three replicas are used in HDFS for fault tolerance, Namenode will select three Datanodes to store a block in the file "a". In Datanode, HDFS should choose a hard disk to each relica of a block. As shown in Figure 1, the last block "k" is stored in disk "2", then the new one "j" will locate at disk "0" according to the round-robin rule.

[1]