Chapter 1

On the Processing of Extreme Scale Datasets in the Geosciences

Sangmi Lee Pallickara, Matthew Malensek, and Shrideep Pallickara
Department of Computer Science
Colorado State University, Fort Collins, USA

1. Introduction

Observational measurements and model output data acquired or generated by the various research areas within the realm of Geosciences (also known as Earth Science) encompass a spatial scale of tens of thousands of kilometers and temporal scales of seconds to millions of years. Here Geosciences refers to the sciences related to the study of atmosphere, hydrosphere, oceans, and biosphere as well as the earth's core. Rapid advances in sensor deployments, computational capacity, and data storage density have been resulted in dramatic increase in the volume and complexity of data in geosciences. Geoscientists now see the data-intensive computing approach as part of their knowledge discovery process alongside traditional theoretical, experimental, and computational archetype [1]. Data-intensive computing poses unique challenges to the geoscience community that are exacerbated by the sheer size of the datasets involved.

Data intensive computing in Geoscience has a unique set of challenges. First many of geo-phenomena are naturally correlated by their geospatial location and time. To cope with increasing volumes and data resolutions in the modeling process, which is spatial and chronological, a wide variety of data analysis technologies have been (or are being) developed. *Cluster analysis* has proven very useful for segmentation, network analysis, change detection, and feature extraction [2]. *Block entropy* can be used as a classifier for a dynamical system. *Spectral methods* are frequently employed for decomposing periodic phenomena. *Artificial neural networks* and model tree ensembles have been used to refine models and to empirically up-scale and extrapolate point measurements [3].

Second, providing access to geo-data for scientists in various domains is not straightforward. Managing high volumes of data from the remote sensors such as satellites or radars poses interesting data management issues such as curation, provenance, metadata creation, and public distribution. It is even more difficult for small datasets from in-situ observational instruments (e.g. measurements for the ecological sampling) to be collected, preserved, distributed, and accessed for further processing [4]. The National Ecological Observatory Network (NEON), a 30-year nationwide study of climate and ecology [5] [6], is in their initial develop-

ment phase to provide infrastructure for distributing observational data collections that enables continental scale analysis and forecasting in ecology and related areas. NEON collects data from electronic sensors mounted on towers (e.g. eddy flux towers) and aquatic array over the continental US. These datasets are useful for monitoring physical and chemical climate properties such as air pollution, carbon concentration, and freshwater sources.

Third, interoperability and data integration is needed in geosciences. There is a wide range of geospatial data repositories, many of which are readily accessible over the Internet. A high degree of interoperability among these systems is imperative. The Open Geospatial Consortium (OGC) has led the effort on standards for geospatial content and services [7]. A key concept underlying this approach is providing a standard way to interoperate with diverse geospatial data management infrastructures and to access heterogeneous forms of geospatial data in a uniform and transparent fashion. Another challenge in data integration is the integration of real-time streaming data, which is becoming a predominant source of geospatial data. OGC's Sensor Web Enablement (SWE) program [8] initiative seeks to provide interoperability between disparate sensors and sensor processing systems by establishing a set of standard protocols to enable a "Sensor Web". In this scenario, sensors in the Web are discoverable, accessible, and usable.

In this chapter, we present current trends and technologies in support of data intensive computing in geosciences. We will focus on the flow of the data, which enables interactions between the data and the computation/analysis. The remainder of this chapter is organized as follows. In Section 2, we discuss major data processes involved in geoscience research projects. This section will cover collection and capture of data from the source, to the visualization and analysis processes. Section 3 will review the geospatial data models and representations. The metadata standards effort will be included as well. In Section 4, we discuss current technologies and systems to manage geospatial data and how they interact with computing resources in various applications.

2. Data Process in Geosciences

Data intensive computing in geosciences involves various processes including data collection, analysis, visualization, and computation. Researchers require a combination of these data processes to achieve their goal. This section describes data processes used for knowledge discovery in the geosciences research based on the characteristics of each process.

2.1 Data Collection from Observational Instruments

Measurement of the natural phenomena is one of the critical tasks in the geosciences. There are two general approaches for performing atmospheric meas-

urements. First, in-situ measurements involve a direct sampling of the atmosphere. Second, remote sensing instruments allow estimation of interesting parameters by measuring changes of related atmospheric radiations. Observations do not have to sample the air directly.

Measurements from in-situ or remote sensing instruments are processed through multiple steps before it is made available to the users. Data processing levels for data products generated as part of a research investigation are categorized [9] as:

- Level 0: Reconstructed and unprocessed instrument payload data at full resolution. This includes any and all communications artifacts.
- Level 1A: Time-referenced and annotated ancillary information that includes radiometric and geometric calibration coefficients besides georeferencing parameters.
- Level 1B: Level 1A data that has been processed while accounting for sensor units.
- Level 2: This includes derived geophysical variables at the same resolution and location as the Level 1 source data.
- Level 3: Variables that are mapped on to uniform space-time grid scales, usually with some completeness and consistency.
- Level 4--Model output or results from analyses of lower level data.

Most users access the datasets in levels 1~4. Since level-4 is the model output, users of our system will access datasets in levels 1~3.

Unlike remote sensing observations, in-situ measurements require additional infrastructure to collect and distribute the data from each of the sites. For example, data collectors such as the WMO's Global Telecommunication Systems(GTS) [10] collect observational data from the global participants and distributes them at the national level. Within the GTS network, Datasets are collected from the World Meteorological Centers (Melbourne, Moscow, and Washington), 15 Regional Telecommunication Hubs (including Beijing, Brasilia, Cairo, and Tokyo), and satellite data centers. These datasets are processed and published by authorized organizations such as the National Centers for Environmental Prediction (NCEP) [11]. NCEP also hosts remote sensing data provided by the National Environmental Satellite and Information Service (NESDIS) and the NEXRAD radar. These datasets can be in different levels of data processing. For example, wind data from NEXRAD radar includes level-2 and level-3 data. NCEP packages datasets based on data similarity (but it still maintains the original structure of reports) and observational cycles. Finally, integrated and encoded datasets are published periodically. Most of these datasets are available through the Internet.

2.2 Data Capture

In geosciences the output of high-throughput computations such as global climate simulations is used for various analysis or visualization steps. These simulations often take hours with thousands of compute nodes. A key challenge is deciding

how to extract the large amount of data being generated by these computations off the compute nodes at runtime and over to the data collecting nodes for further processing such as monitoring, analysis, and archival. This process is referred to as as data capturing and it is important that this has minimal impact on the execution while ensuring reliability.

Parallel file systems improve I/O performance for HPC applications. These include Panasas[12], PVFS [13], Lustre [14], and GPFS [15]. A common goal of these systems is to provide general purpose, multi-user file services. HPC applications can sometimes demand instantaneous and sole access to a large fraction of the parallel file system; this can waste CPU cycles on compute nodes that are waiting for completion of I/O accesses rather than making progress on their scientific simulation. This delay in the shared file system is caused by interference between processes within a single application, or between the applications demanding a higher rate of I/O access [16].

MPI-IP introduced the *abstract device I/O* (ADIO) layer [17] as a way to install system-specific optimizations of the general MPI-IO implementation. ADIO is a user-level parallel I/O interface that provides a portable mechanism to implement a parallel layer within multiple file systems. Collective IO provides a level of data-size driven adaption by aggregating small writes into single, larger writes to obtain better performance. However, this does not address the issue of large writes from all of the processes. The Google File System [18] is focused on higher aggregate throughput, but does not address the interference caused by the shared file system.

In geoscience community, HDF-5 [19], and NetCDF4[20] are popular and these have relied on the underlying IO layer, typically using MPI-IO. PnetCDF [21] provides *subfiling* to address the need to decompose the output to gain greater parallelism.

There have been adaptive I/O methods for improving IO performance by dynamically shifting work from heavily used areas of the storage system to those that are more lightly loaded. ADIOS [22] is one of the adaptive I/O APIs which maintains I/O graphs representing the relationships between nodes for the application, and dynamically schedules the storage based on the I/O cost calculated from the graph. This system has been applied to an astrophysics supernova code, chimera.

2.3 Visualization

The visualization process is closely related to the data model and representation. Visualization algorithms take vast amounts of input produced by the simulation, observation or experiments, and then transform that data into imagery. Modern parallel visualization tools use a data flow network processing design [23]. A dataflow network contains components such as filters, sources, or sinks. A data flow network is composed by relating data objects and components. For example, filters have set of inputs and a set of outputs, both of which are data objects. Sinks have only data inputs and sources have only data outputs. A data flow network is a

pipeline of data with an ordered collection of components. When a pipeline is executed, data comes from the source and flows through filters until it reaches the sink. The majority of data processing in visualization operations are embarrassingly parallel – these process can occur in parallel with no communication between the parallel processes.

VisIt is an example of a visualization application implementing a data flow network with parallel data processing[24]. VisIt is designed for visualization and graphical analysis of terascale scientific simulations such as climate modeling. In VisIt, each component of the data flow network can specify optimizations through *contracts* and communicate with each other to improve performance. For example, many filters specify limits on the amount of data being processed in their contracts. VisIt uses two approaches for rendering. First, surfaces with a relatively small number of geometric primitives are sent to users and rendered locally. Surfaces with a relatively large amount of geometric primitives remain on the server and are rendered in parallel. In terms of the data model, VisIt processes all of the mesh types and field types while tracking and preserving information about data layout and ordering.

2.4 Data Analysis

Data analysis in geosciences involves complex processing and access to a large amount of data. Analysis of atmospheric phenomena involves two important aspects: the physical laws that govern the atmospheric circulation and the spatial and temporal spectra of the atmospheric phenomena [25]. Physical laws indicate how it might be possible to determine one variable from another; for example, analyzing wind from temperature measurements. The governing equations of the atmosphere can be written in terms of independent variables (three dimensional variables for spatial location and time) and dependent variables (temperature, humidity, or chemical species). In general, these equations are nonlinear partial differential equations of the variables described above.

Scientists in geo-sciences have encountered various challenges that increase complexity [26]. First, there is the issue of scale selection . Choosing the best scale for an analysis is an important decision involving experience and also trial-and-error. To achieve this scientists adopt a multi-step approach, in which intermediate results are used to evaluate the next decision in the analysis. The second source of complexity is the uncertainty in the measured data. Data are restricted by trade-offs and practical limitations. A primary objective in data analysis is to figure out random fluctuations from deterministic components. If the data distribution (e.g. Gaussian, exponential, logarithmic, etc.) is known this would is not a problem; however, finding the appropriate data distribution function is a challenging task. Finally, spatial and temporal interactions add to the complexity of the modeling system. For instance, in ecology research, most of the ecosystems are dependent on environmental conditions (e.g. elevation or humidity). Biological variables are altered based on the changing conditions and become space-dependent. If there is

no general dependency in space, a local phenomenon may exist. Space-autocorrelation, which is the phenomena where geospatially neighboring samples are more similar than one would expect from a known ecological condition, is often observed. Similarly, there is the issue of temporal dependence and temporal autocorrelation.

3. Data Models and Representations

As we discussed in section 2, spatial coordinate information and temporal information of the data are keys to organizing geospatial datasets for subsequent processing. Based on the characteristics of the applications and datasets, there have been various data models and types of data representations. In this section, we will review fundamental concepts and techniques for managing geospatial time series datasets in the geosciences.

3.1 Object-based Model and Field-based Model

There have been active efforts in the modeling and representation of geospatial data in GIS (Geographic Information System) based systems. The two most popular approaches are an object-based model and a field-based model [27, 28]. In an object-based model, geographic objects corresponding to real-world entities are defined by a spatial component and a descriptive component. The spatial component is specified by the shape and location of the object in the embedding space. For the object with a given spatial component, a descriptive component provides non-spatial properties in the form of attributes.

In field-based approaches, the space is partitioned into two or multidimensional cells. Each cell has one or more attribute values associated with it and each attribute describes a continuous function in space. An example of a field-based data model is a multispectral or hyperspectral raster imagery obtained from a radar or satellite. In a field-based approach, there is no notion of objects but observations of phenomena described by attribute values (e.g. measurements).

[29] provides a summary of spatial data models used in GIS and example applications as depicted in Table 1. Conversion between models and combining models for more efficient expression of the dataset is common in GIS applications.

Table 1 Geographic Data Models

Data model	Example application
Computer-aided design (CAD)	Automated engineering design and drafting
Graphical (non-topological)	Simple mapping
Image	Image processing and simple grid analysis
Raster/grid	Spatial analysis and modeling, especially in environmental and natural resource applications
Vector/Geo-relational topol-	Many operations on vector geometric fea-

ogy	tures in cartography, socio-economic and resource analysis, and modeling
Network	Network analysis in transportation, hydrology, and utilities
Triangulated irregular network (TIN)	Surface/terrain visualization
Object	Many operations on all types of entities (raster/vector/TIN etc.) in all types of application

3.2 N-dimensional Array Model

Similar to other scientific data, geospatial data is naturally represented by a multi-dimensional, array-based data model [30] [20] [19] Since most commercial databases do not support the array data type, or allow flexible access to large array datasets, there have been many approaches in the geoscience area to cope with this challenge.

First, there are data formats widely used in geosciences: netCDF, [30] and HDFS [19]. These formats follow the Common Data Model (CMD) [31] that represent data as a set of multi-dimensional arrays, with sharable dimensions, and additional metadata attached to individual arrays or the entire file. There are various software available for scientists to access data files that conform to these data formats, including data analysis tools and visualization applications. We will discuss this is section 3.5.

The second approach to dealing with N-dimensional array model for geosciences involves building a scientific storage system supporting multi-dimensional arrays or APIs for accessing multi-dimensional data array stored in existing storage systems. SciDB [32] which is a database management system for scientific research supports a multi-dimensional, nested array model with array cells containing records. SciHadoop provides multi-dimensional array access through their query interface while MapReduce tasks are being executed [33].

3.3 Data Formats: NetCDF, HDF5, and FITS

The Network Common Data Format (NetCDF) [30] is a self-describing data storage format for multi-dimensional data. NetCDF provides a generic and machine-independent interface for storing and accessing scientific data. NetCDF represents data as files containing three properties: dimensions, variables, and attributes. Dimensions are named values used to describe the shape of the variables contained in the file, which could be application-specific attributes such as time, elevation, or position. Dimensions do not necessarily have to be bounded. A variable is rep-

resented as a multi-dimensional array that is a collection of homogeneous values. Finally, attributes describe a variable's metadata or other application-specific information to aid in processing and analysis. NetCDF data format is widely used in scientific data analysis tools (e.g. MATLAB, R) and GIS applications (e.g. ArcGIS), and large-scale simulations.

Hierarchical Data Format 5 (HDF5) [34] was originally developed by the National Center for Supercomputing Applications (NCSA) as a general-purpose, machine-independent scientific data format. HDF5 is the latest version of the format, representing a major architectural redesign over previous versions. An HDF5 file consists of two different components: groups and datasets. Groups allow users and applications to create a hierarchy for data in a tree-based form similar to the POSIX logical file system layout. Datasets are also divided into two parts: a header and a data array. Headers describe the data type, dimensionality, and storage method for the array. They also contain metadata and other information on how the data should be interpreted, which allows HDF files to be self-describing. Data can be stored in a contiguous manner or can be broken into separate pieces of data, called chunks, to improve performance.

Developed in the 1970s for astronomical data, Flexible Image Transport System (FITS) [35] [36] has evolved to support generic scientific datasets. Since FITS is designed for two- or three-dimensional images, it is naturally well-suited for other forms of multi-dimensional scientific data. A FITS file is composed of a number of Header + Data Units (HDUs). The first HDU is an n-dimensional array of pixel data, followed by an arbitrary number of FITS extensions. Extensions include additional n-dimensional images, ASCII tables, or binary tables. Headers contain 80-character key-value entries that describe a FITS file's metadata.

FITS has wide support for programming language bindings, including C, Fortran, Java, Python, and R. Many image-processing applications also support reading image data from FITS files. Since a large use case of FITS is archival storage, additions to the format must not make files produced from a previous version of the format unreadable by newer software implementations.

While the simplicity of the FITS format is a major strength, it may also be a limiting factor for some applications. An 80-character maximum for header records could hinder self-description for some datasets, resulting in additional ASCII or binary data tables needing to accompany data.

4. Data Access in Geosciences

In geosciences data processing schemes are evolving to cope with the large volumes of data generated by observational measurements or from large-scale simulations. Here we discuss approaches taken in the geoscience domain to address this challenge. Our discussion will address distributed data access infrastructure,

GIS, sensor data networks, Grid infrastructure, database management systems, and computational platforms.

4.1 OPeNDAP: Domain Specific Distributed Data Access

The Open Source Project for Network Data Access Protocol (OPeNDAP) [37] provides access to the oceanographic data stored in remote sources. Datasets are stored in several formats at the servers. Users can access and download the data directly from their analysis programs. OPeNDAP provides both server software to make data available to remote users, and client software to access this data. Some of the client software provides data access API libraries for translating between different geospatial models such as NetCDF [30], HDF5[19], JGOFS [38], and others. Although it was originally designed for oceanographic data, it has been used for other geospatial research areas such as atmospheric science and ecology.

The OPeNDAP system provides two types of metadata in the returning dataset. The first type is the syntactic metadata generated from the accompanying data. The second is the semantic metadata that is generated based on the client's understanding of the dataset. OPeNDAP uses these metadata for its translational process and attribute-based search process. The syntactic metadata is useful during translation of the data format and provide a consistent semantic description about the dataset. OPeNDAP also provides an attribute-based search interface to describe the dataset. This includes parameter, range, location, and descriptive search over the metadata.

4.2 PostGIS: GIS approach

As an extension to the PostgreSQL relational database, PostGIS [39] adds support for geospatial datatypes and queries. This permits users and applications to work with geospatial data using standard SQL syntax. PostGIS supports the OpenGIS Simple Features Specification, which defines a set of geometric datatypes and methods for geospatial analysis. This specification provides a level of standardization in the geospatial database field and is supported by industry players including Oracle and ESRI.

To use PostGIS, a new database must be created with the extension enabled. Once created, data can be added to the database using SQL statements. PostGIS also includes a tool that converts ESRI shape files to and from SQL statements. Standard PostgreSQL datatypes are also available in PostGIS-enabled databases. The combination of geospatial datatypes and built-in geospatial methods allow some processing to be offloaded from the clients to the database server itself.

For large datasets, PostgreSQL supports indexing database tables. In the case of multi-dimensional geospatial datatypes, PostGIS uses R-trees based on Generalized Search Tree (GiST) project. R-trees are similar to the common B-tree used in databases and filesystems, and split a geospatial range into a hierarchial set of

bounding boxes [40]. This allows for quick nearest-neighbor and window lookups. One problem with this approach is that the PostgreSQL query planner does not always optimize GiST indexes well, resulting in a query that scans the entire table [39].

PostGIS provides a standardized, SQL-based interface that allows for compatibility with other database systems or data formats. Despite this compatibility, if an application's storage requirements do not fit the OpenGIS Simple Feature Specification, then its data cannot be easily migrated to other systems. Depending on workload, scaling PostgreSQL and PostGIS may also be problematic. To distribute processing across a number of machines, data must be transfered from the database server (or servers) to clients, incurring IO latency costs. To alleviate latencies and distribute data more effectively it is possible to replicate the database or manually split the data across multiple databases, but this complicates queries and the application logic that interacts with the data. However, in cases where geospatial clustering is being performed over objects that span a large geographical area or the entire dataset, the centralized database approach could be more efficient than a distributed (and therefore communication-heavy) approach.

4.3 DataTurbine: Access to the Real-time Sensor Data

Geoscience communities are now actively engaged in large scale sensor-based observational systems. DataTurbine is a real-time data streaming engine [41]. It is an open-source middleware product providing programming abstractions over heterogeneous devices, and integrated network services for managing streaming data. DataTurbine provides a common Application Programming Interface (API) for disparate devices. For example, data streams from accelerometers and video cameras are integrated and managed through a common API, so that users can integrate heterogeneous data streams.

Reliable delivery of sensor data is critical requirement for the large scale sensor network infrastructure because network delays and partitions are common in sensor networks. DataTurbine maintains q ring queue spanning both memory and disk to store the data. It provides a level of reliability that is between plain TCP and application dependent transactional systems. This is also designed to accommodate other functionalities such as real-time data archival and distribution over local and wide area networks. Client applications can use the ring buffers for their on-demand features including data stream subscription, data capture, rewind, and replay.

Routing and topology management in DataTurbine is configurable based on the characteristics of the observational system. For example, a tree-style topology could be applied in situations involving firewalls. Users can configure the sensor nodes behind the firewall as child nodes and define their parent node as nodes located outside the firewall to enable communication between the nodes.

For the geospatial sensor data, DataTurbine provides an online data mapping interface: converting incoming data from sensors to overlays of data. Streamed data is transformed into the KML format and data displays can be layered on the Google Earth interface. This provides a visual interface to the domain experts for easy-to-use navigation over the complex data streams.

4.4 The Earth System Grid: Data Access in Grid Settings

A cyberinfastructure project, the Earth System Grid (ESG) has developed an environment for managing climate data from multiple climate model sources, observation data, and analysis/visualization tools used by the climate research community [42]. Participating institutes publish datasets in ESG. The ESG publication API manages the publishing process. Based on the THREDDS catalog[43], the ESG publication API provides easy-to-use command line tools to package and transfer the newly published datasets. ESG extends the THREDDS metadata specification to organize metadata.

The Berkeley Storage Manager (BeStMan) is an implementation of the Storage Resource Manager (SRM) system used in ESG [44]. BeStMan provides interfaces to various storage systems, including the Mass Storage System (MSSs). High Performance Storage System(HPSS) in LBNL, ORNL, and MSS at NCAR are accessible through a unified ESG gateway portal. BeStMan enables users to access storage systems with different security mechanisms.

Datasets stored at multiple institutes across North America can be moved to the user's site for simulation, analysis or visualization. DataMover-Lite (DML) [42] from the Berkeley Lab is a simple file transfer tool with a graphical user interface that supports various data transfer protocols including http, https, gridftp, ftp, and scp. In ESG data sites are equipped with DML to move files over GridFTP/HTTPS with built-in grid security mechanisms. DML splits files for multiple HTTP connections for faster downloads. For large datasets, the Bulk Data Mover (BDM) [42] provides a scalable data movement solution. BDM manages the transfer of millions of data files those have a total size of 100s of TB reliably.

GridFTP [45] is used as the underlying technology for data download by users, data movement between resources, and data synchronization between replications. Globus Online [46] is a data transfer management software-as-a-service built on top of GridFTP. This software provided by ANL and University of Chicago enables the ESG community to transfer their data. Features include performance monitoring, retrying failed transfers, and automatic fault recovery. Globus Online optimizes the transfer to ensure best performance based on the transfer size and number of files per transfer.

4.5 SciDB: Database Management System for Multi-dimensional data

Instead of building on existing relational databases, SciDB [47] is a different type of database designed specifically for large-scale scientific applications. Storing and processing data from sensor arrays is the primary use case SciDB was designed around, and has been harnessed for appications in seismology, astronomy, and climate research. The creators of SciDB identified three main differences between business data and scientific data: first, in scientific data, the sources of data generally have a location associated with them, which could include adjacency to other sensors or coordinates in space. Second, the data being collected in scientific applications also usually requires complex processing before and/or after storage. Finally, sensors and other scientific instruments generate data on the petabyte-scale, so the database must be able to cope with these massive storage needs.

The logical data storage model in SciDB is different from traditional tabular database systems. SciDB databases consist of n-dimensional arrays of cells. The datatypes held by cells are defined at an array's creation, and they can be either scalars or sub-arrays. These combinations of datatypes and values are called *attributes*. The arrays in SciDB do not necessarily have to be contiguous; cells can be empty, creating a sparse or "jagged" array. Storing data in this manner helps arrays match a variety of scientific data, and also allows data to be grouped in a fashion that is most optimal for processing later. The SciDB storage manager also supports "in situ" data; a large amount of time is spent in the scientific community simply loading and moving data, so SciDB allows manipulating external data that is not part of the database to eliminate load times. While the external data must be in the SciDB format, it is also possible to write adaptors that can allow importing data from formats such as NetCDF or HDF5.

SciDB is a *shared nothing* system, meaning each node in the system runs the SciDB engine and operates independently of other nodes. Collections of data are decomposed into "chunks" and compressed before being stored on a node's local disk. Nodes can be removed or added to the system without affecting other nodes, and data processing operations will continue to run as long as they do not reference data that is unavailable. Upon entry into the system, nodes contact a central catalog to inform the system of their presence, which represents a single point of failure. The catalog is implemented as a PostgreSQL database [48] which also contains information about data chunks stored in the system [47].

To facilitate the division of data, SciDB uses "chunking" and "vertical partitioning." Since cells can contain multiple attributes, vertical partitioning splits the attributes up into their own separate arrays. This division is beneficial because computations often only involve a subset of the overall attributes an array might have. Once the attributes are split, they can be broken up into chunks. Depending on the application, chunks can "overlap," meaning a chunk may contain parts of

adjacent chunks. This behavior is configurable by the database administrator and can greatly reduce communication costs if chunks overlap in a way that fits the processing that will be applied to them, at the cost of using more disk space.

While most relational database tables are updateable by default, SciDB's arrays are not. For many scientific applications, updates are undesirable because keeping record of original results before processing is necessary. To support subsequent processing after the initial import, SciDB records changes to the dataset and exposes this to users in the form of an additional "history" dimension [48]. The history also allows users to change calculations applied to data in a past history snapshot and then have the rest of the values in subsequent processing recalcuated as well.

For processing data, SciDB is modular in nature. It includes a base set of functions for working with data, but also allows user-defined array operators to be written in C++, following a model similar to PostgreSQL's user-defined functions. These array operators can be used to create "enhanced" arrays, where a function is applied to an array and then both the original and enhanced cells can be accessed. Enhanced arrays allow users to access the same data but from different perspectives. Writing array operators facilitates pushing computations to data instead of having to query and transfer information from the database, mitigating the costs of disk and network latencies. In addition, the database's distributed nature also provides parallelism when running array operators across a number of nodes. Extensible language bindings are also provided to allow users to query data with their project's native programming language, if necessary.

4.6 Hadoop MapReduce: Computing Platform

Utilizing the MapReduce paradigm [49] and a form of structured storage is another solution for processing and storing large amounts of geospatial data. As proposed in [50] and [51], Apache Hadoop [52] and HDFS (Hadoop Distributed File System) [53] can be used to execute geospatial queries in parallel.

Many geospatial queries involve computing data surrounding individual points and then processing their combined results, which fits the MapReduce paradigm well. For example, queries often require determining the nearest neighboring points of some arbitrary target point in a geographical area. During the map phase of MapReduce, each data point in the area is distributed to a mapper, which then determines the nearest neighbor of its point. In the reduce phase, any results matching the target point are collected to form a final set of nearest neighbors.

Since standard R-trees represent a global index, they are not suitable for a distributed environment. Wang and Wang [51] propose building indexes for each data file that enters the system to alleviate this problem. Alternatively, instead of

the standard tree-based indexing approaches, Akdogan et al. [50] utilize a flat spatial index using *Voronoi diagrams*. Voronoi diagrams partition a region into a collection of polygons. By removing the tree hierarchy, data coupling is reduced while also allowing the data to be spread evenly across available nodes for better load balancing. This storage model also means that computations can be pushed to the data they operate on rather than requiring programs to request data and have it transferred to them.

While using cloud computing frameworks such as Hadoop can provide benefits for large, data-intensive geospatial applications, there are some downsides to this approach. First, using HDFS results in data files being broken up into pieces called 'splits.' If a query requires data from a neighboring split, an additional MapReduce phase may be required to reconcile results from the two splits. In addition, using a cloud framework for geospatial applications places more burden on the end user to decide how to query and store data, whereas extensions built atop relational databases generally support a common set of geospatial datatypes and procedures that can be exploited with SQL.

4.7 Kepler: Scientific Workflow

While Hadoop and the MapReduce paradigm can provide immense processing benefits for scientific users, there is also a considerable learning curve involved with using the Hadoop framework. The Kepler Project, [54] a scientific processing workflow tool, allows users to create workflows using a graphical user interface. Kepler is a type of "actor-oriented modeling," where actors are components that are designed to perform various processing tasks. In the case of Kepler, MapReduce is implemented as an actor that can be added to workflows.

In a workflow, actors have "ports," which either produce or consume data. Actors generally take data items in, process them, and then pass the results on to the next actor in the workflow. Data may take different paths through the workflow and can execute both serially and in parallel.

Kepler provides a good solution for users wanting to benefit from MapReduce without having to use it for every step in their processing. By default, data files are not stored in HDFS and instead are copied into HDFS from the filesystem before the MapReduce actor runs, so large changes to an existing workflow are not necessary. It is also possible to configure Kepler to use data that is already stored in HDFS, but then other actors would need to support HDFS as well if they need access to the data. The implications of copying files into HDFS before processing are not entirely clear, but could be a large bottleneck when working with massive datasets.

Kepler is particularly well-suited for applications that don't always fit the MapReduce model due to MapReduce being simply one of many possible actors in a workflow. It also makes MapReduce much more accessible for scientific applications.

5. Conclusions

This chapter provided a survey of challenges involved in analyzing geospatial datasets. The proliferation of networked sensors, measurement devices, instruments, and simulations have resulted in large data volumes. Processing such data requires addressing several challenges that include ensuring efficient representations, models, formats, transfers, and computational frameworks.

References

[1] T. Hey, *et al.*, *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Redmond, Washington: Microsoft Corporation, 2009.

[2] F. M. Hoffman, *et al.*, "Multivariate Spatio-Temporal Clustering (MSTC) as a data mining tool for environmental applications," in *the iEMSs Fourth Biennial Meeting: International Congress on Environmental Modelling and Software Society (iEMSs 2008)*, 2008, pp. 1774-1781.

[3] F. M. Hoffman, *et al.*, "Data Mining in Earth System," in *the International Conference on Computational Science (ICCS)*, 2011, pp. 1450-1455.

[4] O. J. Reichman, *et al.* (2011) Challenges and opportunities of open data in ecology. *Science*. 703-705.

[5] M. Keller, *et al.*, "A continental strategy for the National Ecological Observatory Network," *Front. Ecol. Environ Special Issue on Continental-Scale Ecology,* vol. 5, pp. 282-284, 2008.

[6] D. Schimel, *et al.*, "NEON: A hierarchically designed national ecological network," *Front. Ecol. Environ,* vol. 2, 2007.

[7] June, 17, 2011). *The Open Geospatial Consortium (OGC)* Available: http://www.opengeospatial.org

[8] G. Percivall and C. Reed, "OGC Sensor Web Enablement Standards," *Sensors and Transducers Journal,* vol. 71, pp. 698-706, 2006.

[9] *MTPE EOS Reference Handbook* the EOS Project Science Office, code 900, NASA Goddard Space Flight Center, 1995.

[10] *The Global Telecommunication System*. Available: http://www.wmo.int/pages/prog/www/TEM/GTS/index_en.html

[11] *National Center for Environmental Prediction (NCEP)*. Available: http://www.ncep.noaa.gov/

[12] *Panasas: Parallel File System for HPC Storage*. Available: http://www.panasas.com/

[13] M. M. Kuhn, *et al.*, "Dynamic file system semantics to enable metadata optimizations in PVFS," *Concurrency and Computation: Practice and Experience,* vol. 21, 2009.
[14] P. J. Braam, "Lustre: a scalable high-performance file system," 2002.
[15] F. B. Schmuck and R. L. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," in *the Conference on File and Storage Technologies*, 2002, pp. 231-244.
[16] J. Lofstead, *et al.*, "Managing Variability in the IO Performance of Petascale Storage Systems," presented at the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 2010.
[17] M. P. I. Forum, "MPI-2: Extensions to the Message-Passing Interface," 1997.
[18] S. Ghemawat, *et al.*, "The Google File System," *ACM SIGOPS Operating Systems Review,* vol. 37, 2003.
[19] *HDF-5*. Available: http://hdf.ncsa.uiuc.edu/products/hdf5/
[20] *NetCDF4*. Available: http://www.hdfgroup.org/projects/netcdf-4/.
[21] J. Li, *et al.*, "Parallel netCDF: A high-performance scientific I/O interface," in *ACM Supercomputing (SC03)*, 2003.
[22] H. Abbasi, *et al.*, "DataStager: scalable data staging services for petascale applications," in *ACM international Symposium on High Performance Distributed Computing*, 2009.
[23] J. Craig Upson, *et al.*, "The Application Visualization System: A computational environment for scientific visualization," *IEEE Computer Graphics and Applications,* pp. 30-42, 1989.
[24] *VisIt Visualization Tool*. Available: https://wci.llnl.gov/codes/visit/home.html
[25] R. Daley, *Atmospheric Data Analysis*: Cambridge atmospheric and space science series, 1993.
[26] O. Wildi, *Data Analysis in Vegetation Ecology* Willey, 2010.
[27] P. Rigaux, *et al.*, *Spatial Databases with Application to GIS*: Morgan Kaufmann, 2002.
[28] S. Shekhar and S. Chawla, *Spatial Database: A Tour*: Prentice Hall, 2002.
[29] P. Longley, *et al.*, *Geographic Information Systems and Science*, 3 ed.: John Wiley & Sons, 2011.
[30] R. Rew and G. Davis, "NetCDF: an interface for scientific data access," *IEEE Computer Graphics and Applications,* vol. 10, pp. 76-82, 1990.
[31] *Common Data Model*. Available: http://www.unidata.ucar.edu/software/netcdf-java/CDM/
[32] P. Cudre-Mauroux, *et al.*, "A Demonstration of SciDB: A Science-Oriented DBMS," in *the 2009 VLDB Endowment* 2009.
[33] J. Buck, *et al.*, "SciHadoop: Array-based Query Processing in Hadoop," UCSC2011.
[34] (2010, *The HDF Group. Hierarchical data format version 5.* *http://www.hdfgroup.org/HDF5*.

[35] (2011, *FITS Support Office.* http://fits.gsfc.nasa.gov/.
[36] D. C. Wells, *et al.*, "FITS: A Flexible Image Transport System," *Astronomy & Astrophysics,* vol. 44, pp. 363-370, 1981.
[37] P. Cornillon, *et al.*, "OPeNDAP: Accessing data in a distributed, heterogeneous environment," *Data Science Journal,* vol. 2, pp. 164-174, 2003.
[38] D. M. Karl, *et al.*, "Building the long-term picture: U.S. JGOFS Time-series Programs," *Oceanography,* pp. 6-17, 2001.
[39] P. Ramsey, "PostGIS Manual," ed: Refractions Research.
[40] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, ed. Boston, Massachusetts: ACM, 1984, pp. 47-57.
[41] S. Tilak, *et al.*, "The Ring Buffer Network Bus (RBNB) DataTurbine Streaming Data Middleware for Environmental Observing Systems," in *IEEE e-Science*, 2007, pp. 125-133.
[42] D. N. Williams, *et al.*, "The Earth System Grid: Enabling Access to Multi-Model Climate Simulation Data," *Bulletin of the American Meteorological Society,* vol. 90, pp. 195-205, 2009.
[43] B. Domenico, *et al.*, "Thematic Real-time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL," *Journal of Interactivity in Digital Libraries,* vol. 2, 2002.
[44] A. Shoshani, *et al.*, "Storage Resource Managers (SRM) in the Earth System Grid," Earth System Grid2009.
[45] G. Khanna, *et al.*, "A Dynamic Scheduling Approach for Coordinated Wide-Area Data Transfers using GridFTP," in *the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008)*, 2008.
[46] *Globus Online | Reliable File Transfer. No IT Required.* Available: https://www.globusonline.org/
[47] P. G. Brown, "Overview of sciDB: large scale array storage, processing and analysis," in *Proceedings of the 2010 international conference on Management of data*, ed. Indianapolis, Indiana, USA: ACM, 2010, pp. 963-968.
[48] M. S. Mit, *et al.* (2009, *Requirements for Science Data Bases and SciDB*.
[49] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM,* vol. 51, pp. 107-113, 2008.
[50] A. Akdogan, *et al.*, "Voronoi-Based Geospatial Query Processing with MapReduce," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, ed, 2010, pp. 9 -16.
[51] Y. Wang and S. Wang, "Research and implementation on spatial data storage and operation based on Hadoop platform," in *Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on* vol. 2, ed, 2010, pp. 275 -278.
[52] *Apache Hadoop*. Available: http://hadoop.apache.org/
[53] *Hadoop Distributed File System*. Available: http://hadoop.apache.org/hdfs/

[54] J. Wang, *et al.*, "Kepler + Hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems," in *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, ed. Portland, Oregon: ACM, 2009, pp. 12:1-12:8.

Index terms (alphabetically):

Actor-oriented Modeling
atmospheric science
B-tree
chronological data
cloud
curation
data capture
data hosting
data models
Data Turbine
earth science grid
ecology
FITS
geosciences
Geospatial Information Systems
GIS
grid
Hadoop
HDF
HDFS
Kepler
MapReduce
metadata
NetCDF
observational data
OpenDAP
OpenGIS
PostGIS
PostgreSQL
R-tree
raster
SciDB
sensor network
Shared Nothing
simulations
spatial data
topology

vector
Vertical Partitioning
VisIt
visualization
Voronoi Diagram
weather models