International Journal of
Geo-Information

Article

An Array Database Approach for Earth Observation Data Management and Processing

Zhenyu Tan [1], Peng Yue [2,3,*] and Jianya Gong [2]

1 State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, Wuhan 430079, China; tanzhenyu@whu.edu.cn
2 School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China; gongjy@whu.edu.cn
3 Collaborative Innovation Center of Geospatial Technology, Wuhan 430079, China
* Correspondence: pyue@whu.edu.cn

Received: 2 June 2017; Accepted: 17 July 2017; Published: 19 July 2017

Abstract: Over the past few years, Earth Observation (EO) has been continuously generating much spatiotemporal data that serves for societies in resource surveillance, environment protection, and disaster prediction. The proliferation of EO data poses great challenges in current approaches for data management and processing. Nowadays, the Array Database technologies show great promise in managing and processing EO Big Data. This paper suggests storing and processing EO data as multidimensional arrays based on state-of-the-art array database technologies. A multidimensional spatiotemporal array model is proposed for EO data with specific strategies for mapping spatial coordinates to dimensional coordinates in the model transformation. It allows consistent query semantics in databases and improves the in-database computing by adopting unified array models in databases for EO data. Our approach is implemented as an extension to SciDB, an open-source array database. The test shows that it gains much better performance in the computation compared with traditional databases. A forest fire simulation study case is presented to demonstrate how the approach facilitates the EO data management and in-database computation.

Keywords: Earth Observation; multidimensional array; array database; SciDB; Big Data; forest fire simulation

1. Introduction

Earth Observation is a series of activities for collecting, managing, processing, analyzing, and presenting the physical, chemical, and biological information pertaining to the Earth system using remote sensing or other measurement techniques. It has extensive applications and broad prospects in natural resource management and environment monitoring [1]. With the advances in modern sensor technologies, various platforms, such as satellites, planes, and vehicles, have been employed as sensor carriers to gather various observation data, generating a wide range of data types and formats [2]. On the other hand, the improved data resolution in time, space, and spectrum leads to a tremendous growth in EO data size and volume. The information behind these data is more valuable for end users. However, it is estimated that some of the data have never been accessed and processed, causing a waste of resources and incomplete information [3,4]. Hence, how to organize, store, and manage large volumes of EO data and dig out available information from the data requires immediate attention [5,6].

Traditional storage for EO data uses various kinds of files, such as Network Common Data Form (NetCDF) for atmospheric and hydrological sciences, GeoTIFF, and Hierarchical Data Format (HDF) for remote sensing images. These specially-designed data formats work quite well when the amount of data is not very large. However, issues start to arise when data volumes increase gradually. The most obvious problem is that it is not easy to retrieve and query the information needed. To solve this problem, there have been some efforts to provide a more productive environment for EO data storage and processing by leveraging the state-of-the-art multidimensional array databases and High-Performance Computing (HPC) technologies [6]. An array database is designed and implemented as a common database service offering flexible and scalable storage and retrieval on large volumes of multidimensional array data, such as sensor, image, simulation or statistics data [7,8]. It has attracted extensive attention from academic and industry data scientists [7,9,10]. In the geoscience domain, for example, the National Aeronautics and Space Administration (NASA) launched a project—EarthDB to manage tons of Moderate Resolution Imaging Spectroradiometer (MODIS) Level 1B calibrated and georeferenced data by employing an open-source array database named SciDB [11]. The EarthServer is a European Union-funded project to meet the EO Big Data challenges [12]. With all the data stored in Rasdaman array database, it provides a flexible and interoperable processing and analytic service based on Open Geospatial Consortium(OGC) standards. The Australian GeoScience Data Cube (AGDC) is another project that establishes a comprehensive EO Big Data storage and processing framework by employing multidimensional array-based storage and HPC technologies [13]. These are excellent works contributing to the geoscience community.

Based on the works of pioneer contributors, we believe that three aspects should be taken into consideration over EO Big Data storage: (1) how to store large volumes of data in spite of their diverse formats and different observation themes; (2) how to query and extract useful information from massive data archives and contribute to find what users need exactly; (3) how to manage and maintain the growing database with higher efficiency and lower consumption as the velocity and volume of data being generated continue. Correspondingly, the following three guiding strategies are listed. Firstly, common characteristics of EO data should be summarized and concluded to choose a general and consistent storage and management platform. Secondly, easy-to-use query languages or utility tools should be developed to accurately locate the data required. Finally, horizontal scaling should be natively supported, namely, scaling of data storage can be achieved by adding commodity hardware. Overall, the storage system for EO Big Data must be as flexible and scalable as possible.

In this paper, we propose a unified, effective and flexible approach by employing array database technologies to manage and process various EO data. Since most of the EO data, such as remote sensing images, are essentially multidimensional arrays in terms of data structure, it naturally follows to store and manage EO data in an array database, which is specially designed to manipulate data based on semantics of arrays [14]. In addition, array databases offer good extensibility, flexibility, and efficient in-situ data processing capability [15]. Our approach developed a multidimensional spatiotemporal array model. The model is layered on top of common array model and creates a mapping between data coordinate systems and dimensions of arrays, so that data queries within a specific Spatial Reference System (SRS) can be transformed into operations on multidimensional arrays. Basic data processing can be performed directly as in-database computing against the array data models. More sophisticated application analysis can be performed with a high level programming language like Python. In this case, all data operations are performed on array data models, without considering the differences in data types and formats. The model is implemented as an extension to SciDB. The test shows that it gains much better performance in data query and computation compared with traditional databases. A forest fire simulation study case using cellular automation model is presented to demonstrate how the approach facilitates EO data management and in-database computation.

The remainder of this paper is organized as follows. Section 2 introduces the array database technology as well as the proposed modeling method. Section 3 presents the implementation and evaluation. Section 4 shows a case study for forest simulation to go through the approach. Conclusions and future work are described in Section 5.

2. Material and Method

2.1. Material

Various data models have been developed to store and manage Big Data efficiently and effectively [16–18]. Recent years have witnessed the development and prosperity of NoSQL databases. NoSQL databases employ various data models to organize volumes of data. These data models tend to be schema-less, and data are ultimately stored as collections of key-value pairs. NoSQL databases have gained great success in commercial Big Data application, and some attempts have been made to store remote sensing images, LiDAR point clouds and other scientific observation data [19–21]. However, it is not effortless to deal with the array-based EO data in NoSQL databases due to the mismatch of data models and requires transform to a specific data structure in order to load array data into mainstream NoSQL databases.

Meanwhile, the multidimensional array database, as a new rapidly developing technology, came into sight. The array data model matches the nature of the EO data, thus enabling flexible queries based on array dimensions and powerful in-database data processing capabilities. Furthermore, existing array databases support horizontal scaling with just cheap commodity servers set up as clusters to handle ever-increasing data volumes [15]. Data partition among server nodes can be conducted automatically, which is highly desired in the Big Data context.

At present, the most widely-used multidimensional array databases are Rasdaman and SciDB [3,11,22]. They are competing each other, and our initial step is to choose one as a proof-of-concept implementation. The method used in the spatiotemporal array model in this paper, however, is not limited to a specific product. In this paper, SciDB is extended to demonstrate our proposed approach in addressing three considerations for EO data storage in Section 1. SciDB is a scalable, computational, open-source multidimensional array database management system, and has been employed in life science, sensor analytics, financial markets, and other scientific domains [23,24]. It is intended primarily for the storage and management of very large scale array data [10]. In SciDB, every array has a specific schema defining array dimensions and attributes. Attributes stand for the data values stored in the cells of the array, and each cell can contain multiple attributes. SciDB decomposes each array into many subsets with a single attribute, and then each subset is saved as a logical whole [10]. Attributes in SciDB are strong-typed, namely, each attribute in an array shares one specific type. Dimension is the minimum number of coordinates needed to specify a cell in an array. Each dimension is denoted by a name and domain. Currently, SciDB only supports the domain declaring by integer range.

SciDB adopts a shared-nothing, distributed and massively parallel processing architecture [15], shown as Figure 1, which makes it possible to store and access as much data as required by horizontal scaling. A SciDB cluster is composed of a coordinator and a number of database instances running on different data nodes with each node connected with a network [15]. The coordinator is responsible for interacting with all the SciDB instances to handle requests from users. The actual data is partitioned into small chunks and stored in the file systems of distributed server nodes. A PostgreSQL database is employed as a system catalog holding metadata of system and array data. At the front-end, SciDB provides a higher-level Array Query Language (AQL) and Array Functional Language (AFL) for data query and manipulation. Moreover, a HTTP-based client called shim and other Application Programming Interfaces (APIs), such as Python and R APIs, are also provided to access and query data from databases.

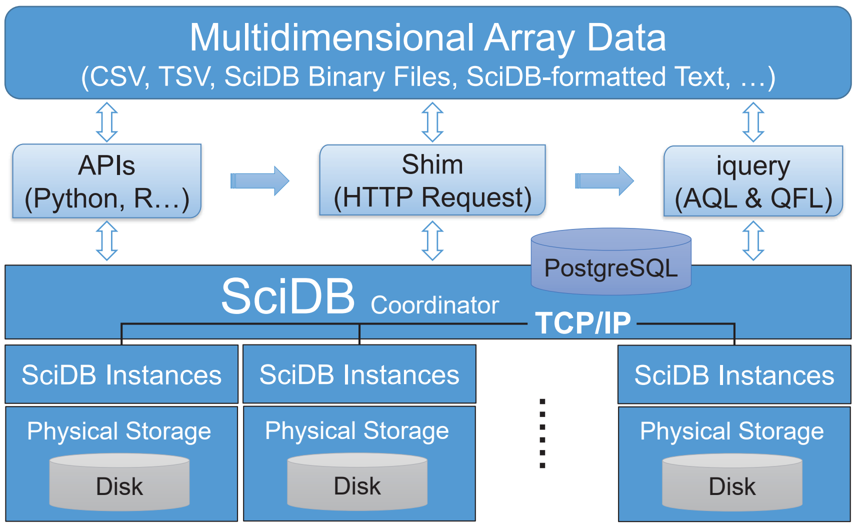


Figure 1. System architecture of SciDB.

2.2. Method

A key issue in applying the multidimensional array model for EO data management is to develop a spatiotemporal array model. What makes EO data different from other scientific data is the spatial reference information attached with data. In our approach, observation data in the array database are organized as multidimensional spatiotemporal data cubes. Geospatial dimensions, usually denoting by latitude and longitude, are indispensable for the multidimensional data cubes. For long-time series data, time dimension is also needed. Other dimensions can be added according to the actual data in practice. Array attributes are used to hold the observation values. For example, a single remote sensing image can be regarded as a data cube with three dimensions $(lat, long, band)$ and each cell in the array contains a value representing ground surface reflectance. Optionally, it can also be regarded as a two-dimensional array with multi-attributes, and each attribute contains one band value. Figure 2 presents an overall picture of our proposed approach for EO data management. In addition to the data model, the geospatial metadata are records instead of arrays. So that geospatial metadata could still take advantage of traditional Relational Database Management Systems (RDBMSs). Specifically, EO metadata are persisted in relational tables, while actual observation data are stored in the array database. Thus the approach adopts a hybrid storage mechanism by combining RDBMS and array database. Metadata and observation data are linked by unique array identifiers. General speaking, the hybrid storage mechanism, multidimensional spatiotemporal data cubes, and mapping strategy between array coordinates and spatial coordinates are key parts in our array database approach.

The mapping strategy must keep the spatiotemporal semantics in the array database model when loading EO data into databases. An array coordinate system is usually defined by its dimensions. A dimension has its starting and ending coordinates, which are denoted by integers with interval identically equal to one. However, EO data are often presented in a specific SRS, and the coordinates are often not denoted by integers. Hence, a mapping between array coordinates and geospatial coordinates is proposed to solve the mismatch of coordinate systems.

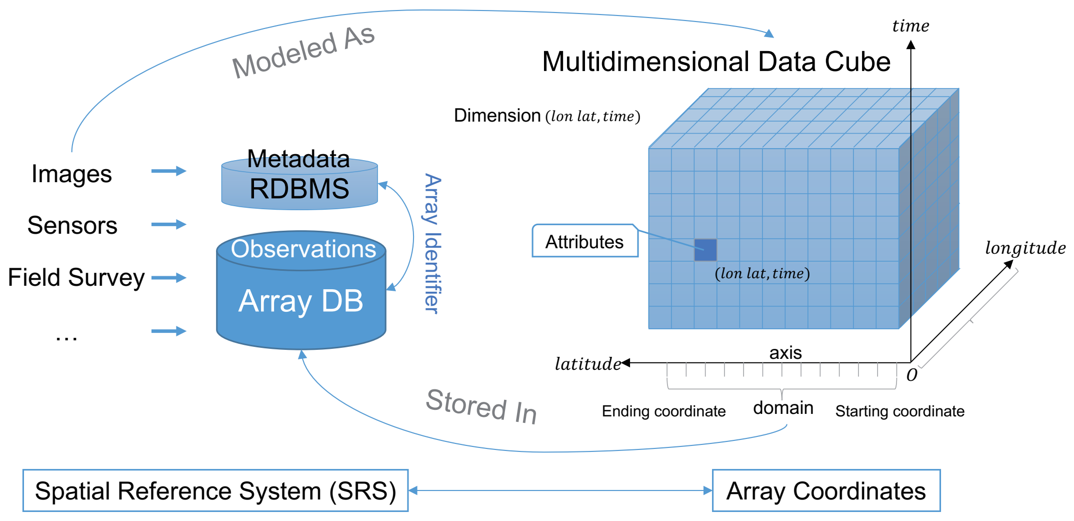
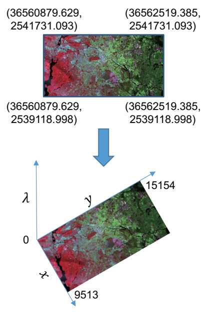
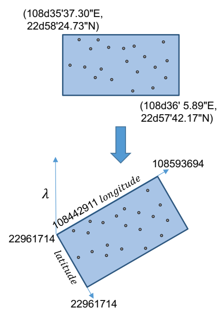
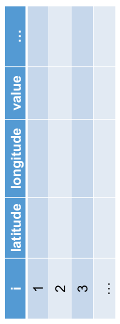


Figure 2. The array database approach for EO data management.

Here discusses the mapping strategy for different kinds of EO data. In terms of the continuity of observed area, EO data can be classified as continuous and discrete. Continuous data usually cover a large contiguous area representing a continuous natural phenomenon or condition, such as remote sensing images. Discrete data are usually discrete observation points, such as in-situ sensor observations. We adopt a generic six-parameters affine transformation model [25], as shown in Equation (1), to establish the mapping . In this equation, (x_{geo}, y_{geo}) is the projected coordinate, (x_{pixel}, y_{line}) is the raster coordinate staring from $(0,0)$, and the rest are determined transformation parameters. With this transformation, user queries in a specific SRS can be mapped into operations in array coordinate system. It works for the data with either projected coordinate system or geographic coordinate system. For discrete data, there are two ways to model data as arrays. One way is to take geospatial coordinates as dimensions, and use the parallel and equally spaced attitude and longitude grids to organize the observation data with each data item located in a single cell. We can scale up the attitude and longitude values by multiplying a multiple of ten to make them integers. The range between scaled minimum and maximum integers can be served as array dimension range. The enlargement factor depends on the data resolution. In this way, it will form a sparse array with many cells having no values. Another way is taking geospatial coordinates as attributes of arrays. In this way, it will form one-dimensional arrays if not taking time dimension into consideration. And this one dimension counts items of observation. In an array database, these two-form arrays can be easily transformed to each other. Table 1 summarizes the different situations to model EO data in order to load them into array databases.

$$\begin{cases} x_{geo} = x_0 + a_{11} \cdot x_{pixel} + a_{12} \cdot y_{line} \\ y_{geo} = y_0 + a_{21} \cdot x_{pixel} + a_{22} \cdot y_{line} \end{cases} \tag{1}$$

Table 1. EO data organization in array database.

	Earth Observation Data		
Category	**Continuous**	**Discrete**	
	Multidimensional spatiotemporal arrays		
Data structure	Dimensions: $(x, y, \cdots)$ Attributes: (observed values. . .)	Dimensions: $(lat, long, \cdots)$ Attributes: (observed values. . .)	Dimensions: $(i, \cdots)$ Attributes: (lat, long, observed values. . .)
Mapping strategy	Six-parameter affine transformation	Scale up the attitude and longitude values by multiplying by ten	/
Spatial dimension denotation	 (36560879.629, 2541731.093) (36562519.385, 2541731.093) (36560879.629, 2539118.998) (36562519.385, 2539118.998) λ y 15154 0 x 9513	 (108d35'37.30"E, 22d58'24.73"N) (108d36' 5.89"E, 22d57'42.17"N) 108593694 λ 108442911 longitude 22961714 latitude 22961714	 i latitude longitude value … 1 2 3 ⋮

In contrast to the array data model, we organize EO metadata by themes and sources, and build series of new relation tables to store the metadata. Figure 3 gives the Entity-Relation (ER) model represented as a Unified Modeling Language (UML) diagram. The relation *spatial_dim* holds the mapping information between array coordinates and geospatial coordinates by using a six-parameter affine transformation. The relation *spatial_ref* records the projection information. And the relation *time_dim* is the description information on the time dimension with the begin timestamp and interval. This is a light weight design, which includes essential relations of the data theme and sources, as well as spatial and time dimensions. Metadata tables like the *dataset1_meta* and *dataset2_meta* are added for each dataset. It is identified by the *array_name* field, which refers to the arrays in array databases, so that arrays can be linked up with their spatiotemporal semantics and related metadata.

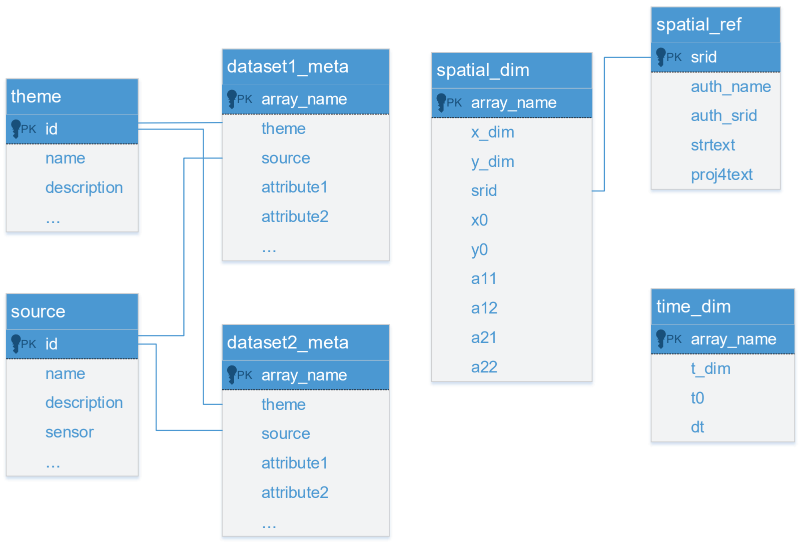


Figure 3. ER model for EO metadata.

After loading data into the molded arrays, traditional window queries, and various pixel-based calculations can be translated into operations performed on array dimensions and attributes. As a result, array databases shield the differences among various data sources, by providing a unified multidimensional data cube perspective for end users. It also presents the geospatial specialty of EO data using array dimensions so that EO data can be correlated with other data in a unified data retrieval and analysis framework. End users can now focus on the design of algorithms and analysis models for scientific experiments and simulations, without caring about various data formats, types, and data transformations.

3. Implementation and Evaluation

3.1. Implementation

A prototype system is built for EO data management and processing. Figure 4 gives an overview of the system architecture. The key to the prototype system is the multidimensional spatiotemporal array data cube. Different types of EO data can be easily integrated into an array database in a uniform way. At the bottom layer, EO data can be transformed and loaded into SciDB. By setting up a data cluster with servers, incoming data will be distributed in different nodes. As data volumes grow increasingly, more data nodes can be dynamically added. EO metadata are organized by data themes and sources. For a specific theme and data source, there is a specific metadata table with specific fields. Moreover, there are additional tables for the definitions of arrays, spatiotemporal dimensions, and SRSs. All of this information is persisted in the PostgreSQL database. Users can handily query the metadata with standard Structured Query Language (SQL), and retrieve observation data with SciDB AQL and AFL.

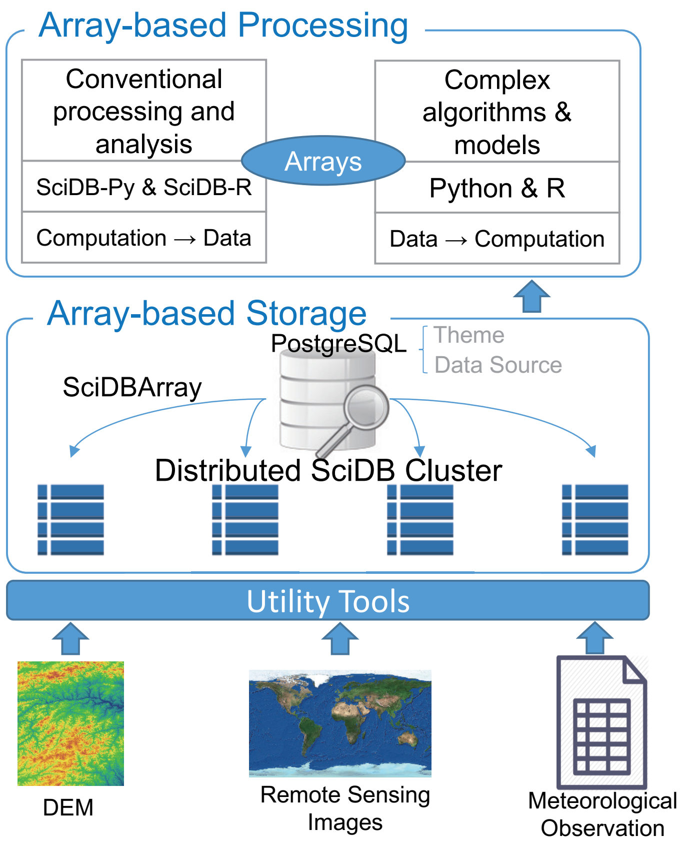


Figure 4. System architecture for EO data management and processing.

When it comes to the data processing, SciDB offers Python and R interfaces to operate the data in the array database. Python and R are programming languages widely used in scientific computing with native support for multidimensional array data structure. However, the computing capabilities of the programming languages are limited to the main memory of computers and data volumes that exceed available memory cannot be processed. To overcome this limitation, SciDB-Py and SciDB-R can be used, which are well-designed interfaces to access and manipulate the data residing in a SciDB database [26]. The programming function calls will eventually be transformed into the SciDB operations performed in distributed data nodes. For the analysis with massive data volumes and simple calculations, such as conventional linear algebra and matrix operations, we can just use AFL to do the computation. This can be called migration from computation to data, compared to conventional data migration computing. We refer it as in-database computing. Considering those analyses with less data volumes and complex processing logics, data from SciDB can still be migrated into customized analytical models following the conventional way. This is because currently SciDB cannot support some complex in-database processing. However, if advanced analyses are needed in databases, we could implement them as user-defined functions (UDF) with C++ APIs.

3.2. Comparison with Related Software Solutions for EO Data Cubes

There are some existing projects on EO data cubes. Among them, the Rasdaman and AGDC are well-known efforts. There are also some similarities and differences between our approach and the existing work. This section compares the implementation strategies among Rasdaman, AGDC and our approach. Rasdaman implements the capabilities for accessing EO data by an application called Petascope [12]. Petascope fully implements the OGC Web Coverage Service (WCS) and Web Coverage Processing Service (WCPS) standards and offers a standard interface for on-line data access and processing [27]. Geospatial metadata of EO data is modeled according to OGC GML Coverages (GMLCOV) specification and persisted in PostgreSQL database. Spatiotemporal coordinates are parsed by an component named Semantic Coordinate Reference System (SECORE) [28]. Currently, Petascope only supports raster data whose axes are aligned with the axes of the Coordinate Reference System (CRS) defined in SECORE, so spatial coordinates of each raster cell can be calculated by the original point coordinate and an offset vector which determines the geometric distance between grid points along its axes [29]. Using Petascope and SECORE, all the WCS or WCPS requests from clients can be directly translated into *rasql* (Rasdaman Query Language) and executed by Rasdaman.

AGDC is a another platform for managing EO Big Data following the data cube approach. It provides a High-Performance-Computing and High-Performance-Data (HPC-HPD) environment for efficient data storage and processing. This system consists of four layers: data acquisition and inflow, data cube infrastructure, data and application platform, as well as user interface and application layer [13]. Source datasets go through series of preprocessing and flow into the spatiotemporal data cube. The data cube infrastructure offers a suite of utilities to partition data and save them as array-based NetCDF files. AGDC also provides spatial indices and multidimensional array-based query interfaces. Data and application platform utilizes the HPC technologies to perform geoscience analysis and simulation. The top layer provides client-side interactions and visualization. It has been shown that the AGDC-enabled platform can enhance the efficiency of geoprocessing and analysis [30].

Table 2 gives a comparison between three solutions for EO data cubes. In summary, the SciDB-based approach is implemented as a database plugin. After loading data into database, users can retrieve observation data as well as their metadata with SciDB AQL/AFL. Moreover, there are plenty of built-in functions that can be combined to perform advanced processing. This approach can be extended to add more geoprocessing functionalities by SciDB UDFs. Rasdaman approach is achieved by a web application providing a standard data access and processing interface. It ensures a good interoperability level based on OGC standards. AGDC organizes data as spatiotemporal data cube and offers rapid processing by enabling HPC technologies. Compared with the first two approaches, the storage of AGDC relies on traditional file-based storage. In contrast, array database approaches could provide more flexible queries with their elaborately designed query languages. The advantages of AGDC platform is that it provides a comprehensive HPC-HPD environment based on the unified array-based data structure and HPC technology.

Table 2. A comparisons between solutions for EO data cubes.

	SciDB	**Rasdaman**	**AGDC**
Implementation	Database plugin	Web application	From scratch
Key technologies	SciDB database	Rasdaman database	HPC+HPD (NetCDF)
Array database supported	Yes	Yes	No
Metadata storage	PostgreSQL (themes, data sources, customized tables)	PostgreSQL (fixed schema, GMLCOV model)	PostgreSQL (fixed schema, JSONB)
Coordinate mapping	Affine transformation	GMLCOV model	/
Array-based interface provided	Yes	Yes	Yes
User interaction	AFL, AQL, C++/Python/R API	WCS & WCPS, rasql, C++/Java/R/ JavaScript API	Python API
Processing pattern	In-database	In-database	HPC
Advantages	Extensible	Standard	Comprehensive

3.3. Performance Evaluation

This section evaluates the performance of SciDB for EO data management. Testing is designed from two aspects. First, a performance test is conducted between SciDB and PostgreSQL database. PostgreSQL is a representative of open-source relational database approach and has been widely used for managing geospatial data with the PostGIS extension [31]. Second, we compare the performance between a single node and multi-node cluster for SciDB. Each comparison includes three types of tests: data import, typical queries, and data processing. Table 3 lists the testing environmental and test data. Table 4 lists testing items as well as examples.

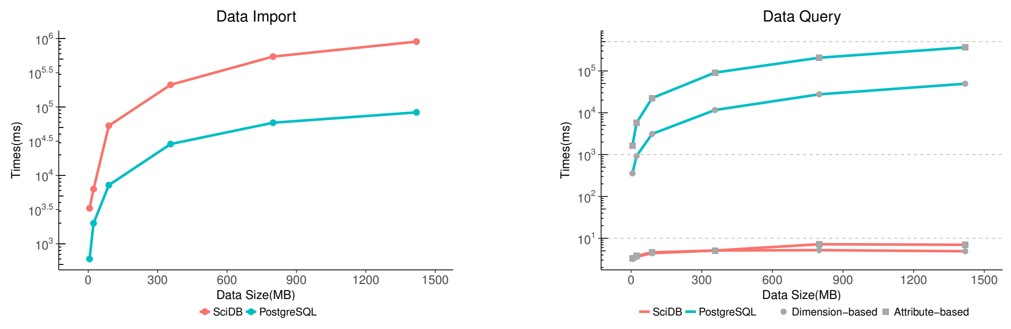
Table 3. Performance evaluation environment.

Item	**Single Node**	**Cluster**
Hardware	One server node	Two server nodes
	Every node shares the same hardware configurations OS: Ubuntu 14.04 x64 CPU: Intel(R) Xeon(R) CPU E5-2692 v2 @ 2.20 GHz, 12 cores RAM: 31.00 GB	
Software	SciDB Community Edition 15.12 PostgreSQL 9.3.16 + PostGIS 2.1.2	
Data	Six remote sensing images:	
	5.6 MB (Size: 975 × 993 × 3)	23 MB (Size: 1950 × 1987 × 3)
	89 MB (Size: 3900 × 3975× 3)	356 MB (Size: 7801 × 7951 × 3)
	799 MB (Size: 11701 × 11926 × 3)	1419 MB (Size: 15602 × 15902 × 3)

Table 4. Performance evaluation items.

Item	Description	Example	SciDB Functions
Data import	Load raw data into database		
Typical queries	Queries based on dimension	Retrieve data according to a specific geographic extent	*between()*
	Queries based on attribute	Retrieve data where its cell values lie within a given range	*filter()*
In-database processing	Aggregation operation	Calculate the sum of an array	*aggregate(), sum()*
	Arithmetic operation	Add cell values of one array to other array	$+, -, *, /$
	Dimension transformation	Change the band of an image as other dimension	*redimension(), unfold()*
	Comprehensive processing	Water extraction	*apply()*, $+, /$
		Mean filtering	*window(), agv()*

In PostgreSQL database, SQL is used to perform test items listed in Table 4. In SciDB database, its built-in operators, functions, and aggregates are used in combination. Figure 5 is the result of comparison between SciDB and PostgreSQL databases in a single node. From the result, we can tell that PostgreSQL shows more efficiency in geospatial raster data loading, while SciDB is significantly faster than PostgreSQL in data query and processing. In particular, the cost time of query and arithmetic computation in SciDB is relatively stable and accomplishes almost in a few milliseconds. This demonstrates that SciDB is highly efficient in array data queries and linear computation. Figure 6 is the result of performance tests between a single node and a cluster of SciDB. The multi-node cluster shows much more efficient for the time-consuming processing, such as the comprehensive processing example shown in Figure 6f. However, a single-node SciDB is a little bit faster for the query and processing sometimes. This is particularly prominent when the process can be finished in a few milliseconds, since in such simple cases, the communication delay in a SciDB cluster outperforms the computing time.



(**a**) (**b**)

Figure 5. *Cont.*

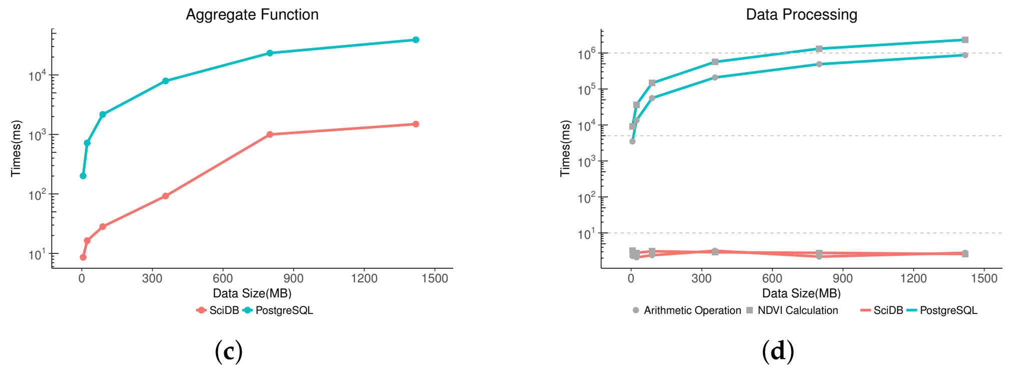


Figure 5. Performance comparison between SciDB and PostgreSQL. (**a**) Data import; (**b**) Data query; (**c**) Aggregate function; (**d**) Data processing.

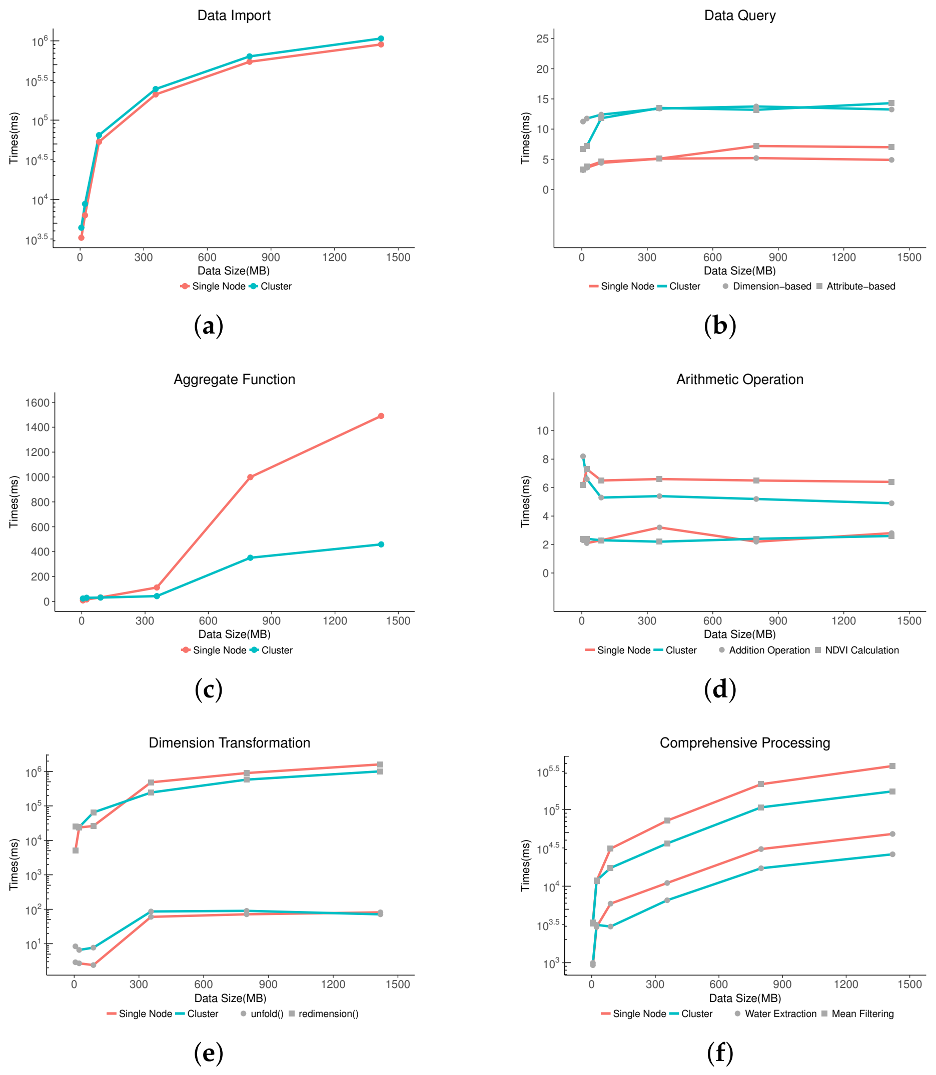


Figure 6. Performance comparison between a single node and a cluster of SciDB. (**a**) Data import; (**b**) Data query; (**c**) Aggregate function; (**d**) Arithmetic operation; (**e**) Dimension transformation; (**f**) Comprehensive processing.

4. Case Study

4.1. Forest Fire Simulation Model

To prove the feasibility of our approach, a case study on forest-fire simulation is demonstrated. The fire model is borrowed from [32], which was designed based on the field research in Greater Khingan Range, China. There are many factors that contribute to a forest fire. This model selectively takes temperature, wind, land cover, and topographic slope as key factors. Its fundamental equation is as follow:

$$R = R_0 \cdot K_s \cdot K_w \cdot K_f \tag{2}$$

where R_0 is the initial fire spearing speed; K_s, K_w and K_f denote the adjusted coefficients of land cover, wind speed and topographic slope respectively. R_0 can be calculated with the regression equation $R_0 = aT + bV - c$, where T and V denote surface temperature and wind speed respectively; a and b are regression coefficients. As for K_s, the modified mapping relations between land cover categories and K_s values are adopted according to the field research. And this equation evolves into the following form with further research in [33]:

$$R = R_0 \cdot K_s \cdot K_w \cdot K_f = R_0 \cdot K_s \cdot e^{0.1783V} \cdot e^{3.533(\tan\varphi)^{1.2}} \tag{3}$$

where φ is the absolute value of slope.

In this paper, we combine this forest-fire model with Cellular Automata (CA) theory to simulate the spreading of a forest fire in Greater Khingan Range. CA is a complex discrete dynamical system in time and space [34]. The state of each cell in CA is synchronized and updated under some specific local rules and constraints. The cell space of our CA system is formed by the two dimensional latitude and longitude grid and a Moore neighborhood is applied. The next burning state of a cell is influenced by the current state of itself as well as eight neighbors. The rules performed on each cell are as follows: if $V_{k,l}^t$ is the spreading speed from cell (k, l) to its neighbors at time t, then after a period of Δt, the cell state transition function can be expressed as follows:

$$\begin{aligned} S_{i,j}^{t+1} = S_{i,j}^t &+ \frac{(V_{i-1,j}^t + V_{i1,j-1}^t + V_{i+1,j}^t + V_{i,j+1}^t)\Delta t}{a} \\ &+ \frac{[(V_{i-1,j-1}^t)^2 + (V_{i-1,j+1}^t)^2 + (V_{i+1,j-1}^t)^2 + (V_{i+1,j+1}^t)^2]\Delta t}{2a^2} \end{aligned} \tag{4}$$

If $S_{i,j}^t < 1$, then cell (i, j) is partially burning and cannot spread throughout its neighbors. If $S_{i,j}^t > 1$, then cell (i, j) is completely burned and begins to spread over its eight neighbor cells. The cell state ranges from 0 to 1.

4.2. Data Storage Model

According to the influence factors of forest fire, MODIS, Modern-Era Retrospective analysis for Research and Applications (MERRA) and Shuttle Radar Topography Mission (SRTM) datasets are collected for the simulation. Table 5 gives the details of these data. As in the aforementioned approach, metadata are organized into PostgreSQL relation tables, and actual observation data are stored in SciDB. Since all the datasets are continuous data, affine transformation is used for coordinate mapping. The array schema defined in SciDB reflects how the data are stored. In general, the schema comprises the definition of attributes and dimensions. An attribute is denoted by name and data type. A dimension is defined by its name, a minimum and a maximum value range, chuck length and chunk overlapping [35]. Taking the MODIS Land Cover products as an example, we take spatial coordinates

(x, y) and time as three dimensions and five land cover type layers as attributes to build array in SciDB. Table 6 gives the array schemas of our data.

Table 5. Data items and their influence factors.

Data Source	Product Name	Fire Growth Factor	Data Size
MODIS	Land Cover Dynamics Yearly L3 Global 1 km	Land cover	273.7 MB
MODIS + MERRA	Land Surface Temperature /Emissivity Daily L3 Global 1 km MERRA-2 ($0.5° \times 0.5°$)	Surface temperature	MODIS: 13.5 MB MERRA: 397 MB
MERRA	MERRA-2 ($0.5° \times 0.5°$)	Wind speed	61.3 MB
SRTM	SRTM Non-Void Filled, 90 m	Topographic slope	117.4 MB

Table 6. Array schemas of each data type in SciDB.

Array for MODIS Land Cover:

```
MCD12Q1_A2013001_h25v03_051<Land_Cover_Type_1:uint8,Land_Cover_Type_2:uint8,
    Land_Cover_Type_3:uint8,Land_Cover_Type_4:uint8,Land_Cover_Type_5:uint8>
    [y=0:2399,4096,0, x=0:2399,4096,0, t=0:*,0,0]
```

Array for MERRA2:

```
MERRA2_400_inst1_2d_lfo_Nx_20131201<PS:float,QLML:float,SPEEDLML,float,
    TLML:float>[y=0:360,2048,0,x=0:575,2048,0,t=0:23,0,0]
```

Array for SRTM DEM:

```
srtm_61_02 <val:int16 > [y=0:5999,2048,0,x=0:5999,2048,0]
```

4.3. Simulation Walk Through

The CA-based forest fire simulation is implemented by integration of SciDB and Python. Figure 7 is the execution flow of whole procedure. First, algorithm described in [36] is used to derive slope from SRTM Digital Elevation Model (DEM) data. The calculation is achieved in SciDB database. A built-in $window()$ operator is used to produce a new array where each output cell is some aggregate calculated over a window around the corresponding cell in the source array [35]. The aggregate function to calculate the slope of each cell is implemented by user-defined functions (UDFs). Second, raw data are retrieved from SciDB within a specific latitude and longitude extent in the form of *numpy.ndarray* object in Python. Each array contains the information of one influence factor. Third, every array needs to be interpolated to obtain the same size covering the whole study area. Finally, the aforementioned fire model is applied to each cell to dynamically simulate the spreading of fire. The initial fire points are generated randomly, and then the spreading speed of the burning point is calculated according to Equation (3), looping through each cell to calculate the fire state according to Equation (4). With time elapsing, cell states are updating correspondingly.

The forest fire simulation result shown in Figure 8 is the overlapping of fire states and MODIS land cover data. Initial burning points are selected randomly as in Figure 8a. From Figure 8b to Figure 8d, it exhibits the process of fire spreading as time goes by. The color ramp representing the state of burning fire. Crimson indicates it is completely burned with cell value equal to 1, pink indicates just starting to burn with value equal to 0. We can see that the fire is spreading evenly in surrounding areas of burning points. This is because, as noted in Table 5 above, the spatial resolution of collected data differs a lot. The MERRA datasets are coarse-grained, and MODIS land surface temperature data have large pieces of "no data" values. Despite the interpolation process, it cannot improve the spatial resolutions of data. The simulation result can be definitely effected.

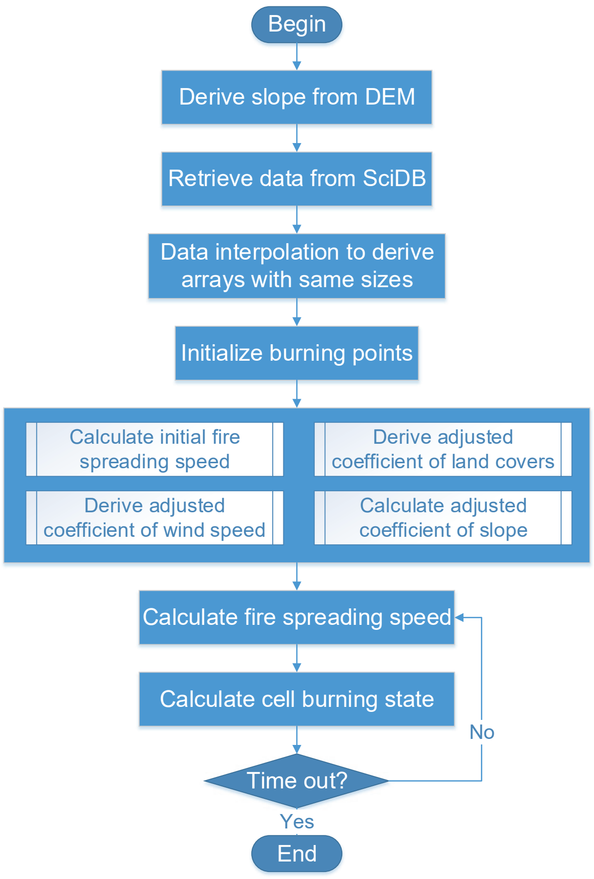

Begin
Derive slope from DEM
Retrieve data from SciDB
Data interpolation to derive arrays with same sizes
Initialize burning points
Calculate initial fire spreading speed
Derive adjusted coefficient of land covers
Derive adjusted coefficient of wind speed
Calculate adjusted coefficient of slope
Calculate fire spreading speed
Calculate cell burning state
No
Time out?
Yes
End

Figure 7. Forest fire simulation processing flow.

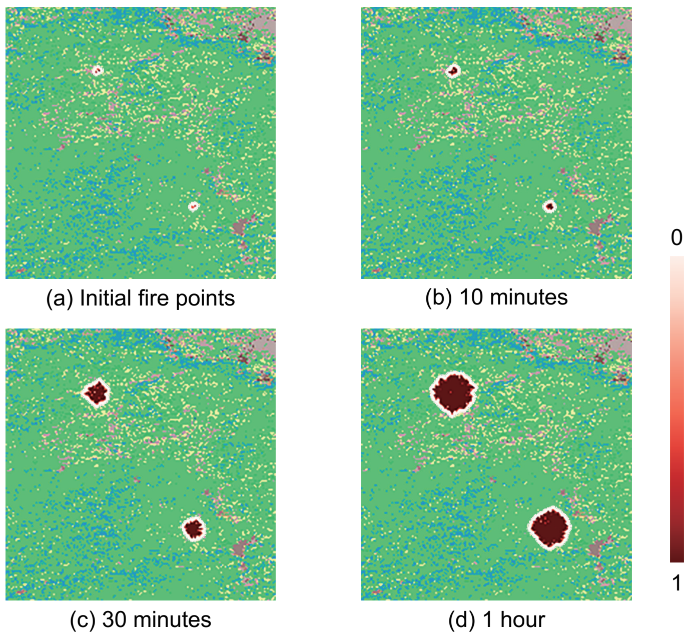


Figure 8. Forest fire simulation result.

4.4. Result Analysis

Generally, the case study of the forest fire simulation mainly includes three steps: loading EO data into database, data preprocessing and model simulation. The case is to show how in-database computing can be combined with complex analysis and simulation. It took about 380 s to load all the data listed in Table 5 into SciDB. Next, it took about 72 s to perform the database reprocessing, e.g., the slope computation, while the same process took 526 s in a s single-machine environment with ArcGIS software. The distributed in-database computing in SciDB shows much higher efficiency than the traditional processing pattern. At present, however, not all the sub-processes can be achieved in SciDB, and sometimes it requires extensive work to accomplish a new customized function. Therefore, the simulation part is implemented by pure Python. The simulation took around 30 min for a one-hour real fire scenario with a 10-s interval. The execution time is highly related to time interval and study area. Based the performance evaluation and the simulation, it can be shown that SciDB is efficient for large data array linear computation. The performance could be further improved in a distributed cluster. One significant characteristics of big geospatial data is the variety [18]. The variety is not limited to different sources of data and various data formats, but also is highlighted by different processing methods. In this case, tradition simulation computation is combined with in-database computation. When dealing with traditional feedback based iterations in simulation, the data volume is small while time-step dependent loop computation is often needed [37]. In this case, traditional in-memory based scientific computing is more appropriate without the need to interacting databases at each time step. Instead, the in-database processing, e.g., the slope computation, is performed in the initialization phase of the simulation. This allows to take the best of both approaches for better performance in a complex analysis.

We can benefit from using this approach for EO applications. The data used in this case study come from different data sources, and are in different formats. Traditionally, different software may be used to process the data, such as coordinate transformation, clipping, and format transformation, to integrate different data sources into the same model. With the proposed spatiotemporal multidimensional array database approach, all that are needed is to load data into an array database, and users can query data from the database and perform conventional processing with built-in and customized functions. Moreover, by using Python programming, with its good support for multidimensional array data structure, geospatial users just need to deal with arrays. Through SciDB-Py API, data can be retrieved by the given geographic extent, then all the data from different observation themes are converted into Python arrays. In this way, different data can be seamlessly integrated into the same model. Furthermore, the implementation could be improved by more optimization strategies, such as decomposing arrays into small parts and adopting multi-thread processing mechanism.

In the approach, the traditional heterogeneity problems, such as different data sources, data formats, and data projections, are now addressed in a unified array database model. All types of EO data can be stored and retrieved in consistent array semantics. Basic data processing can be performed in the database with its build-in operators, aggregates and functions. More complex operations can be implemented with UDFs. In-situ processing in array databases is particularly efficient for the Big Data. And the database cluster with the horizontal scaling feature can be scaled to support high performance data management, as well as computing. It is better to provide more powerful in-database processing utilities, so that we could explore scientific workflow approaches for database side scripting instead of user-defined functions. The scientific workflow approach could allow more flexible and customized processing and improve the processing capabilities of databases.

5. Conclusions and Future Work

This paper presents how to leverage array database technologies for EO data management and processing. It suggests a hybrid storage mechanism, multidimensional spatiotemporal data cubes, and mapping strategy for digesting EO data in array databases. A prototype system using SciDB is presented. The performance tests show that SciDB is quite scalable and it has the potential for EO Big

Data management. Moreover, the array-based processing by integrating SciDB and Python works well for scientific computing and simulation. The forest fire case study illustrates the feasibility to manage EO data in an array database, and to integrate different data sources in one scientific model. The array database approach can be applied in other domains and benefit in-situ processing in databases.

The mapping from EO data to spatiotemporal arrays still needs some improvement. The array coordinates are usually detonated by integers, while spatial coordinates by doubles. The transformation may cause some precision problem. While sometimes we could trade precision for performance, there is some balance here. In addition, in the future it could be possible to explore the possibility of other geocoding mechanisms such as GeoHash as the spatial dimension for EO data, in particular discrete data. SciDB is a promising array database product, and it is being continually developed. Currently it only supports a limited number of functions. Fortunately, SciDB is open source. Users can create new functions to meet their needs. It is better to provide some good extensibility mechanisms, so that complex processing can be easily implemented. So far, the ecosystem around SciDB is not rich, lacking flexible tools for EO data import and domain-specific analysis. Related tools still need to be improved. Furthermore, we want to explore the possibility to implement some common geographic processing model within SciDB and take full advantages of its in-database processing capabilities.

Acknowledgments: We appreciate the three anonymous reviewers for their very constructive comments that helped improve the quality of the paper. The work was supported by Major State Research Development Program of China (No. 2017YFB0504103), National Natural Science Foundation of China (No. 91438203), Nature Science Foundation Innovation Group Project of Hubei Province, China (No. 2016CFA003), Hubei Science and Technology Support Program in China (No. 2014BAA087), Program for New Century Excellent Talents in University in China (No. NCET-13-0435), and Wuhan Yellow Crane Talents (Science) Program.

Author Contributions: Peng Yue and Jianya Gong designed the method and conceived the experiments; Zhenyu Tan performed the experiments and analyzed the data; Zhenyu Tan and Peng Yue wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guo, H.; Liu, Z.; Jiang, H.; Wang, C.; Liu, J.; Liang, D. Big earth data: A new challenge and opportunity for Digital Earth's development. *Int. J. Digit. Earth* **2017**, *10*, 1–12.
2. Di, L.; Moe, K.; van Zyl, T.L. Earth observation sensor web: An overview. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2010**, *3*, 415–417.
3. Karantzalos, K.; Bliziotis, D.; Karmas, A. A scalable geospatial web service for near real-time, high-resolution land cover mapping. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 4665–4674.
4. Ma, Y.; Wu, H.; Wang, L.; Huang, B.; Ranjan, R.; Zomaya, A.; Jie, W. Remote sensing big data computing: Challenges and opportunities. *Future Gener. Comput. Syst.* **2015**, *51*, 47–60.
5. Lee, J.G.; Kang, M. Geospatial big data: Challenges and opportunities. *Big Data Res.* **2015**, *2*, 74–81.
6. Yue, P.; Ramachandran, R.; Baumann, P.; Khalsa, S.J.S.; Deng, M.; Jiang, L. Recent activities in earth data science. *IEEE Geosci. Remote Sens. Mag.* **2016**, *4*, 84–89.
7. Yue, P.; Baumann, P.; Bugbee, K.; Jiang, L. Towards intelligent GIServices. *Earth Sci. Inform.* **2015**, *8*, 463–481.
8. Tan, Z.; Yue, P. A comparative analysis to the array database technology and its use in flexible VCI derivation. In Proceedings of the 2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics), Tianjin, China, 18–20 July 2016; pp. 1–5.
9. Baumann, P.; Dehmel, A.; Furtado, P.; Ritsch, R.; Widmann, N. The multidimensional database system rasDaMan. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD '98), Seattle, WA, USA, 1–4 June 1998; pp. 575–577.
10. Brown, P.G. Overview of sciDB: Large scale array storage, processing and analysis. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10), Indianapolis, IN, USA, 6–10 June 2010; pp. 963–968.
11. Planthaber, G.; Stonebraker, M.; Frew, J. EarthDB: Scalable analysis of MODIS data using SciDB. In Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data. ACM (BigSpatial '12), Redondo Beach, CA, USA, 6 November 2012; pp. 11–19.

12. Baumann, P.; Mazzetti, P.; Ungar, J.; Barbera, R.; Barboni, D.; Beccati, A.; Bigagli, L.; Boldrini, E.; Bruno, R.; Calanducci, A.; et al. Big data analytics for earth sciences: The earth server approach. *Int. J. Digit. Earth* **2016**, *9*, 3–29.
13. Lewis, A.; Oliver, S.; Lymburner, L.; Evans, B.; Wyborn, L.; Mueller, N.; Raevksi, G.; Hooke, J.; Woodcock, R.; Sixsmith, J.; et al. The Australian geoscience data cube—Foundations and lessons learned. *Remote Sens. Environ.* **2017**, doi:10.1016/j.rse.2017.03.015.
14. Baumann, P.; Holsten, S. A comparative analysis of array models for databases. In Proceedings of the International Conferences on Database Theory and Application, Bio-Science and Bio-Technology, DTA/BSBT 2010, Jeju Island, Korea, 13–15 December 2010.
15. Paradigm4. *The Architecture and Motivation for SciDB Paper Is Out*; Technical Report; Paradigm4: Waltham, MA, USA, 2016.
16. Han, J.; E, H.; Le, G.; Du, J. Survey on NoSQL database. In Proceedings of the 2011 6th International Conference on Pervasive Computing and Applications, Port Elizabeth, South Africa, 26–28 October 2011; pp. 363–366.
17. Harrison, G. *Next Generation Databases: NoSQLand Big Data*; Apress: New York, NY, USA, 2015.
18. Yue, P.; Zhang, C.; Zhang, M.; Zhai, X.; Jiang, L. An SDI approach for big data analytics: The case on sensor web event detection and geoprocessing workflow. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 4720–4728.
19. Xiao, Z.; Liu, Y. Remote sensing image database based on NOSQL database. In Proceedings of the 2011 19th International Conference on Geoinformatics, Shanghai, China, 24–26 June 2011; pp. 1–5.
20. Liu, Y.; Chen, B.; He, W.; Fang, Y. Massive image data management using HBase and MapReduce. In Proceedings of the 2013 21st International Conference on Geoinformatics, Kaifeng, China, 20–22 June 2013; pp. 1–5.
21. Wang, W.; Hu, Q. The method of cloudizing storing unstructured LiDAR point cloud data by MongoDB. In Proceedings of the 2014 22nd International Conference on Geoinformatics, Kaohsiung, Taiwan, 25–27 June 2014; pp. 1–5.
22. Appel, M.; Lahn, F.; Pebesma, E.; Buytaert, W.; Moulds, S. Scalable earth-observation analytics for geoscientists: Spacetime extensions to the array database SciDB. In Proceedings of the EGU General Assembly Conference, Vienna, Austria, 17–22 April 2016; Volume 18, p. 11780.
23. Cudre-Mauroux, P.; Kimura, H.; Lim, K.T.; Rogers, J.; Simakov, R.; Soroush, E.; Velikhov, P.; Wang, D.L.; Balazinska, M.; Becla, J.; et al. A demonstration of SciDB: A science-oriented DBMS. *Proc. VLDB Endow.* **2009**, *2*, 1534–1537.
24. Brown, P. A survey of scientific applications using SciDB. In Proceedings of the New England Database Day 2015, Cambridge, MA, USA, 30 January 2015.
25. Warmerdam, F. The geospatial data abstraction library. In *Open Source Approaches in Spatial Data Handling*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 87–104.
26. Stonebraker, M.; Brown, P.; Zhang, D.; Becla, J. SciDB: A database management system for applications with complex analytics. *Comput. Sci. Eng.* **2013**, *15*, 54–62.
27. Aiordăchioaie, A.; Baumann, P. PetaScope: An open-source implementation of the OGC WCS Geo service standards suite. In *Proceedings of the Scientific and Statistical Database Management: 22nd International Conference, SSDBM 2010, Heidelberg, Germany, 30 June–2 July 2010*; Gertz, M., Ludäscher, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 160–168.
28. Baumann, P.; Campalani, P.; Yu, J.; Misev, D. Finding my CRS: A systematic way of identifying CRSs. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12), Redondo Beach, CA, USA, 6–9 November 2012; pp. 71–78.
29. Rasdaman GmbH. Petascope User Guide. Available online: http://www.rasdaman.org/wiki/PetascopeUserGuide (accessed on 28 June 2017).
30. Lewis, A.; Lymburner, L.; Purss, M.B.J.; Brooke, B.; Evans, B.; Ip, A.; Dekker, A.G.; Irons, J.R.; Minchin, S.; Mueller, N.; et al. Rapid, high-resolution detection of environmental change over continental scales from satellite data—The Earth Observation Data Cube. *Int. J. Digit. Earth* **2016**, *9*, 106–111.
31. Obe, R.O.; Hsu, L.S. *PostGIS in Action*, 2nd ed.; Manning Publications Co.: Greenwich, CT, USA, 2015.
32. Wang, Z. Current forest fire danger rating system. *J. Nat. Disasters* **1992**, *3*, 39–44.

33. Mao, X. The influence of wind and relief on the speed of the forest fire spreanding. *Q. J. Appl. Meteorol.* **1993**, *1*, 014.
34. Wolfram, S. Universality and complexity in cellular automata. *Physica D* **1984**, *10*, 1–35.
35. Paradigm4 Inc. SciDB Reference Guide. Available online: https://paradigm4.atlassian.net/wiki/display/ESD/SciDB+Database+Arrays (accessed on 10 May 2017).
36. De Smith, M.J.; Goodchild, M.F.; Longley, P.A. *Geospatial Analysis*; Troubador Publishing: Leicester, UK, 2009.
37. Yue, P.; Zhang, M.; Tan, Z. A geoprocessing workflow system for environmental monitoring and integrated modelling. *Environ. Model. Softw.* **2015**, *69*, 128–140.