Managing large multidimensional hydrologic datasets: A case study comparing NetCDF and SciDB

Haicheng Liu, Peter van Oosterom, Theo Tijssen, Tom Commandeur and Wen Wang

ABSTRACT

Management of large hydrologic datasets including storage, structuring, clustering, indexing, and query is one of the crucial challenges in the era of big data. This research originates from a specific problem: time series extraction at specific locations takes a long time when a large multidimensional (MD) dataset is stored in the NetCDF classic or the 64-bit offset format. The essence of this issue lies in the contiguous storage structure adopted by NetCDF. In this research, NetCDF file-based solutions and a MD array database management system applying a chunked storage structure are benchmarked to determine the best solution for storing and querying large MD hydrologic datasets. Expert consultancy was conducted to establish benchmark sets, with the HydroNET-4 system being utilized to provide the benchmark environment. In the final benchmark tests, the effect of data storage configurations, elaborating chunk size, dimension order (spatio-temporal clustering) and compression on the query performance, is explored. Results indicate that for big hydrologic MD data management, the properly chunked NetCDF-4 solution without compression is, in general, more efficient than the SciDB DBMS. However, benefits of a DBMS should not be neglected, for example, the integration with other data types, smart caching strategies, transaction support, scalability, and out-of-the-box support for parallelization.

Key words | chunked storage structure, hydrologic dataset, NetCDF, SciDB

Haicheng Liu (corresponding author)
Peter van Oosterom
Theo Tijssen
Tom Commandeur
Faculty of Architecture and the Built Environment,
Delft University of Technology,
Julianalaan 134, 2628 BL Delft,
The Netherlands
E-mail: *h.liu-6@tudelft.nl*

Wen Wang
State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering,
Hohai University,
Nanjing, 210098,
China

INTRODUCTION

In the hydrologic domain, original data collection techniques with improved spatio-temporal accuracy, for instance radar systems, are becoming increasingly prevalent. Meanwhile, new sensor platforms such as multispectral lidar and citizen-supplied observations provide more possibilities to collect data. All kinds of simulation models never stop running to produce essential results for decision-making. However, large quantities of these data are normally stored and distributed in diverse formats, which causes professionals inconvenience in effectively preparing information for different applications. Formats widely used include Hierarchical Data Format (HDF), Network Common Data Form (NetCDF) and Gridded Binary (GRIB) which were originally designed for meteorological purposes. Among them, NetCDF is notable for its simple data model, ease of use, portability, and strong user support infrastructure (Rew *et al.* 2006). It is widely applied to record and distribute meteorological, oceanographic as well as hydrologic observations or simulation results.

However, according to the practical experience of industrial engineers, traditional NetCDF solutions perform inefficiently in retrieving time series from large spatio-temporal datasets. This is caused by the contiguous storage structure that it utilizes to store variable values such as rainfall and temperature. Basically, NetCDF stores spatial grids as a one-dimensional array according to a row-majored

doi: 10.2166/hydro.2018.136

order (Figure 1(a)–1(c)). Thus, to query the value in a particular cell, the cell position in the one-dimensional array is calculated. Extraction of a time series (Figure 1(a)) thus becomes expensive due to accessing individual cell values, which are widely spread over the disk, each moment in its own one-dimensional array (Figure 1(c)). Alternatively, it is possible to store a time series for every location as a one-dimensional vector in NetCDF, but then retrieving a complete spatial grid at one moment in time becomes the problem.

Regarding management of large numbers of multidimensional (MD) array datasets, it is natural to adopt a database management system (DBMS) solution as it could provide an easy-to-use interface and fine scalability. In practice, organizations can have a range of data types, and a standardized and generic DBMS solution would be preferable for combining various datasets in different hydrologic applications, e.g., flood decision support system (Abebe & Price 2005), water quality management (Trepanier *et al.* 2006), and lake monitoring (Crétaux *et al.* 2011). In addition, DBMS can offer rich functionalities thanks to ad hoc query support and declarative programming models. Most modern DBMSs also support automatic parallelization in query execution. The MD array DBMS is optimized further to

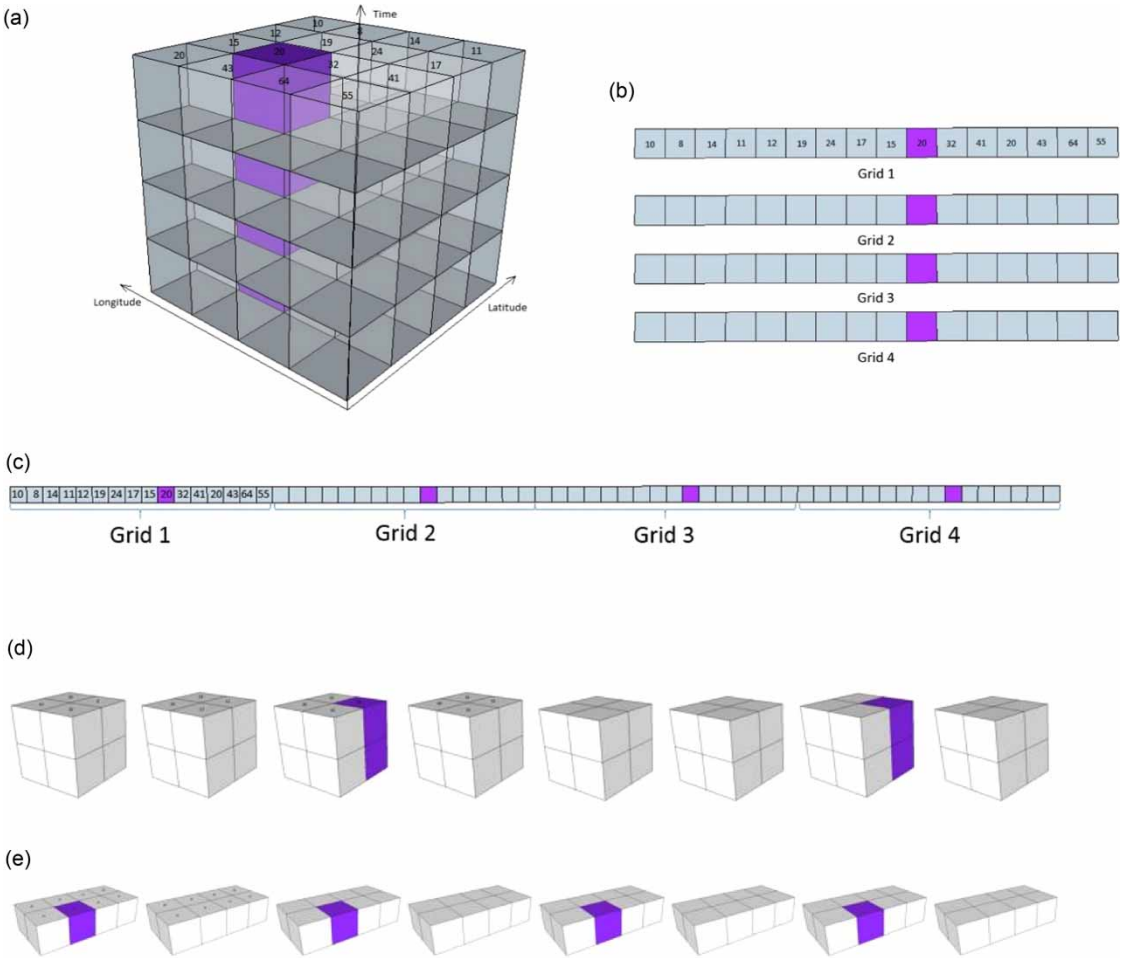

Figure 1 | The contiguous storage and the chunked storage of a three-dimensional precipitation dataset. (a) The three-dimensional precipitation sample and a time series inside at a specific location. Only cell values in the first grid (layer) are shown. (b) The encoding of each grid in a row-majored order. (c) Concatenation of all grids into a one-dimensional array. (d) Storage of the precipitation dataset with the chunked storage structure while the chunk size is 2 (longitude) × 2 (latitude) × 2 (time). Each chunk is also stored as a one-dimensional array on the disk, but chunks are independent from each other. (e) The chunked storage of the precipitation dataset using 2 (longitude) × 4 (latitude) × 1 (time) as the chunk size.

support array data management as well. It can specify metadata (Pedersen & Jensen 2001) and support query features like versioning and overlapping (Colliat 1996; Brown 2010). It employs the chunked storage structure (Figure 1(d) and 1(e)) which divides a whole dataset into separate chunks with specified chunk sizes (Baumann *et al.* 1998; Brown 2010). Then MD array DBMSs apply specific array addressing and relative offset calculation to index cells, which achieves high query efficiency (Colliat 1996).

Previous studies compare relational DBMS MySQL (without specific MD array support) and MD array DBMS SciDB on managing big astronomical datasets (Cudre-Mauroux *et al.* 2012). Researchers cooperate with astronomical experts to propose scientific data processing benchmark and then perform tests. Their final results show that SciDB performs one or two orders of magnitude faster for most queries than MySQL. There is also research about the comparison between a standard relational DBMS and NetCDF classic solution on executing four aggregations (sum, average, minimum, and maximum) and three index queries (first, middle, and last element) (Cohen *et al.* 2006). Results demonstrate the superiority of NetCDF solution over a relational DBMS (without MD array support) for large datasets. However, for publications reviewed until now, no research has been conducted to investigate the query efficiency of NetCDF solutions compared to MD array DBMSs for hydrologic applications.

Hence, this research is aimed at investigating whether a MD array DBMS can achieve better performance in processing queries on large MD hydrologic datasets than classic non-chunked NetCDF and chunked NetCDF-4 file-based solutions. NetCDF-4 is a later version introduced to the NetCDF model, introducing the concept of chunking to this file format. Based on the prior research (Liu *et al.* 2016), enhanced benchmark tests are executed, elaborating benchmark establishment, MD array DBMS selection, construction of the benchmark test environment, benchmark execution, and results analysis. Several solutions will be benchmarked in this research:

- Contiguous data in NetCDF 64-bit offset format without compression
- Chunked data in NetCDF-4 format without compression
- Chunked data in NetCDF-4 format with compression
- Chunked data in MD array DBMS without compression
- Chunked data in MD array DBMS with compression.

DATA AND QUERY

In order to assess various solutions for managing MD array data, it is important to clearly specify representative types of data and corresponding series of queries. The performance of different solutions depends on how similar the data are organized and stored compared to the selection requested. Devising an optimal solution is a challenge in which a proper balance has to be found, which can then be evaluated with the benchmark. Therefore, several experts are first interviewed (Table 1) to better understand the nature of the data and the queries. These experts are deliberately selected from different countries and different hydrologic occupations to increase the comprehensiveness of the benchmark.

To clearly define the various queries, a categorization of query types is performed. Based on previous related studies (Cohen *et al.* 2006; Su & Agrawal 2012) and practical experience, four main query classes are determined (Table 2). In most cases, users first select the area of interest and a certain period of time (Class A), i.e., range/box selection according to spatio-temporal values, retrieving dependent variable values. Class D is also frequently implemented by hydrologists. Class B and C are sometimes executed but they are not the most common types. Therefore, Class A and D get priority in the benchmark.

Considering all the requirements for datasets, i.e., data size, dimension, and accessibility, two datasets (Table 3) have been selected for testing. Dataset **MPE** (Multi-Sensor Precipitation Estimate) stores the rainfall rate data processed from raw satellite observations. It is a combination of measurements from a passive microwave imager and infrared data from EUMETSAT geostationary satellites (Heinemann *et al.* 2002). Hydrologic Research BV can provide records for more than two years, and the total amount of data is larger than 4.18 TB (2 years × 365 days × 24 hours × 250 MB). Dataset **GEFS** (Global Ensemble Forecast System) is calculated from a global weather forecast system, i.e., the GFS from the National Centers for Environmental Prediction (NCEP) (Ashrit *et al.* 2013). The forecast dimension refers to the time steps simulated in the model,

Table 1 | Queries and datasets collected by consultancy

Expert	Datasets	Queries
Dr. Ir. Steele-Dunne (Delft University of Technology)	Remote sensing data products, such as TRMM, GRACE and SMOS	Detection of observations, i.e., selecting non-null variable values Quality control, selecting values from a variable grid according to the quality grid
Prof. Wang (Hohai University)	Observation records collected from spot gauges	Accumulating daily variable values to yearly based values for a dataset
	Satellite imageries	Averaging extreme value of a variable per year for a dataset
	Data products of remote sensing, such as MODIS NDVI, and NPP data	Quality control. No-value area should be below 10% for the whole spatial grid
Ir. Commandeur (Hydrologic Research B.V.)	Time series recorded by gauges including precipitation, discharge, and temperature	Statistical operations, e.g., 'sum' and 'average'
	Precipitation data derived from Doppler radars	Maximum selection
	Radar precipitation data calibrated using records from gauges. The data contain two spatial dimensions and one temporal dimension	Sub-selections according to spatial or temporal dimension from multidimensional datasets
	Processed satellite data such as precipitation and soil moisture data. They incorporate two spatial dimensions and one temporal dimension	Subtracting one grid of one dataset from the other grid of another dataset
	Forecast datasets calculated from models, e.g., the Global Ensemble Forecast System. Data have five dimensions, longitude, latitude, time, ensemble, and model run	Detecting observations from data captured by orbiting satellites
	Orbiting satellite observations in swathes	Calculating the percentile curve for forecast datasets
Ing. Van der Wielen (Hydrologic B.V.)	Time series from gauges	Sum of time series from gauges
	Radar data with two spatial dimensions and one temporal dimension	Extracting time series from grids
	Results computed from hydrologic models. The output has four dimensions, longitude, latitude, time, and model run date	Subtraction of accumulated data, e.g., precipitation
	Forecast data from models	Combining data of two grids with different cell sizes. Intersection and multiplication are needed
		Assigning color to polygons according to theme values by intersecting polygons with grids
		Data pyramid calculation and query on the pyramid
Ir. Villa Real (IBM Brazil)	Topographic and land cover data	Selecting data from one variable grid according to a NODATA grid
	Precipitation rasters calculated from forecast models	

Table 2 | Classes of queries collected from consultancy

Query/operation class	
A. Selection based on spatial/temporal extent	B. Selection based on the variable value
C. Spatial join/combination/masking operations	D. Mathematical calculation such as *sum*, *avg*

while model run represents the wall-clock time to run the model in reality. The 20 ensembles simulate 20 different initial conditions as the input for the forecast model. This is done to decrease the uncertainty of the forecast as later the percentile and ensemble mean are derived.

According to the expert interviews, the most important query types for the MPE and GEFS datasets have been

Table 3 | Datasets for benchmarking

Dataset	MPE	GEFS
Variables	Rainfall rate	Temperature 2 m above ground; Maximum temperature 2 m above ground; Total precipitation; etc.
Dimension count	3	5
Dimension span	x, y, time; (4,000,4,000,4)	Longitude, latitude, forecast, ensemble, model run; (360,181,40,20,1)
Temporal resolution	15 minutes	6 hours
Spatial resolution and coverage	0.03 degree (3.3 km); 1/3 world	1 degree (111 km); global
Single file size	250 MB	1.55 GB
Data format	NetCDF 64-bit offset	NetCDF 64-bit offset

selected. In plain English, the queries for benchmarking are as follows.

MPE:

(Q1) Selection based on spatial dimensions for Delft
(Q2) Selection based on spatial dimensions for north Netherlands
(Q3) Extraction of time series for a single location in the Indian Ocean
(Q4) Computation of the monthly average value for the Netherlands.

GEFS:

(Q1) Selection of total precipitation for all ensembles at Delft (a single spot) for one forecast step
(Q2) Selection of total precipitation for all ensembles at Delft for 40 forecast steps
(Q3) Selection of the 80th percentile from all ensembles at Delft for 40 forecast steps
(Q4) Selection of the mean of 20 ensembles of total precipitation for 40 forecast steps in the Netherlands (a bounding box).

SELECTION OF MD ARRAY DBMS

In this research, MD array DBMS is defined as a DBMS of which the abstract model for data management and query supports operations on MD arrays consisting of dimensions and attributes. A dimension may represent a real physical dimension such as latitude, longitude, height, and time. It can also be used to index other quantities, for example, the model run number. Attributes are the 'variables' in NetCDF, representing the information of interests like precipitation, rainfall rate, and evapotranspiration. Traditional relational DBMSs employ tables as the abstract data model for data management. Although columns in the table can be regarded as different dimensions or attributes, the role or importance of each column is equal, while in the MD array DBMS, dimensions are used to organize attribute data and the focus lies in the attributes. Queries to a MD array DBMS will never only select dimension values, but rather use the dimensional values to specify the range of area/time of interest for which the actual attribute values are requested.

To determine a suitable MD array DBMS for benchmarking, popular solutions are compared. These include Rasdaman (Baumann *et al.* 1998), SciDB (Stonebraker *et al.* 2011), MonetDB (Idreos *et al.* 2012; Gonçalves *et al.* 2016), Essbase (Oracle 2008), Intersystems Caché (InterSystems 2017), and Oracle spatial (Oracle 2014; Xie 2016). Among them, Rasdaman and SciDB provide sufficient documentation for study and research, and the source code is accessible online, which is crucial for exploring more details of data structures. To analyze the management and the query performance of both solutions specifically, nine relevant criteria (Table 4) are established for comparison (Appendix, Table A1, available with the online version of this paper). The SciDB community solution provides lossless compression which is a significant feature for big data processing. Also, SciDB 14.3 has an active user forum, which can help in understanding

Table 4 | Criteria for comparing rasdaman and SciDB

Criterion	Interpretation
License	Open source product is preferable because, on the one hand, more details of storage and query execution can be acquired from source code, and on the other hand, no additional budget is needed for buying licenses
Implementation of the MD array storage	Chunked storage structure is employed by most MD array DBMSs. But details for real implementation such as spatial clustering and indexing can influence the query performance as well, which should be considered
Compression support	Compression is of interest in this research because for hydrologic datasets, usually a large amount of zero values or null values can exist and it is deduced that compression should thus have a notable influence on the query efficiency
Parallelization	Parallel processing capability is a general technique dealing with big data to improve the query performance
.Net C# API	The HydroNet-4 system which is used for benchmarking is constructed on the .Net platform. To keep consistency, it is preferable that the DBMS can be accessed through C# APIs
Query language	Query language of the DBMS should have a simple yet powerful interface, which provides plenty of functionalities for communicating with the DBMS
Spatial computing capability	The DBMS is expected to be able to perform spatial operations such as intersection, distance calculation, and projection conversion
NetCDF importer	This is needed to load NetCDF files directly into the DBMS. Datasets originally in different formats can be converted to NetCDF formats inside the HydroNET-4 system
Maintenance	The DBMS should provide maintenance to fix bugs, improve and add new features if they are required by large numbers of users. As well, the consultancy supporting of the DBMS should also be sufficient, which can benefit this research

benchmark results. It is a bottom-up designed array DBMS from the physical storage layer to logical access layer. Also, it utilizes the chunked storage structure with its native binary format.

BENCHMARK ENVIRONMENT

The benchmark test environment is constructed on the HydroNET-4 system (Reichard *et al.* 2014), which is running on Microsoft Windows. However, SciDB (version 14.3) is only available on Linux. Thus, a SciDB connector has been developed such that the query performance of both NetCDF and SciDB can be assessed in the same HydroNET frontend system (Figure 2). The hardware remains the same.

As is shown in Figure 2, a query starts at a web client which sends an HTTP POST request with a JSON object containing essential parameters for executing the query. The data access API receives the request and parses it. Then the API communicates with the catalog DBMS to retrieve the data from the specific data stores through connectors. After data extraction, the result is returned as a binary HydroNET in-memory object to HydroNET-4. The NetCDF connector used to read files in 64-bit offset format is based on HydroNetCDF library developed and optimized by the company, and the NetCDF-4 connector is developed using the standard library NetCDF-4.1.3. The SciDB connector is constructed on 'shim', a basic SciDB client that exposes limited SciDB functionalities through a simple HTTP API. All connectors can communicate with the *Processor*. For benchmarking, the *Processor* is utilized to realize complex queries such as aggregation and percentile calculation for the NetCDF solutions, while SciDB processes all calculations itself and communicates with the API directly. The elapsed real time spent for executing a query, i.e., the benchmark result, can be measured in the HydroNET-4 frontend. The whole benchmark environment is installed on a Dell Inc. OptiPlex 745 server in the company, Hydrologic BV. The server has one Intel processor with two cores, 6,600 at 2.4 GHz, 4 × 2 GB DDR2 RAM, 3 TB SATA 5,400 rpm Western Digital hard disk. As a large

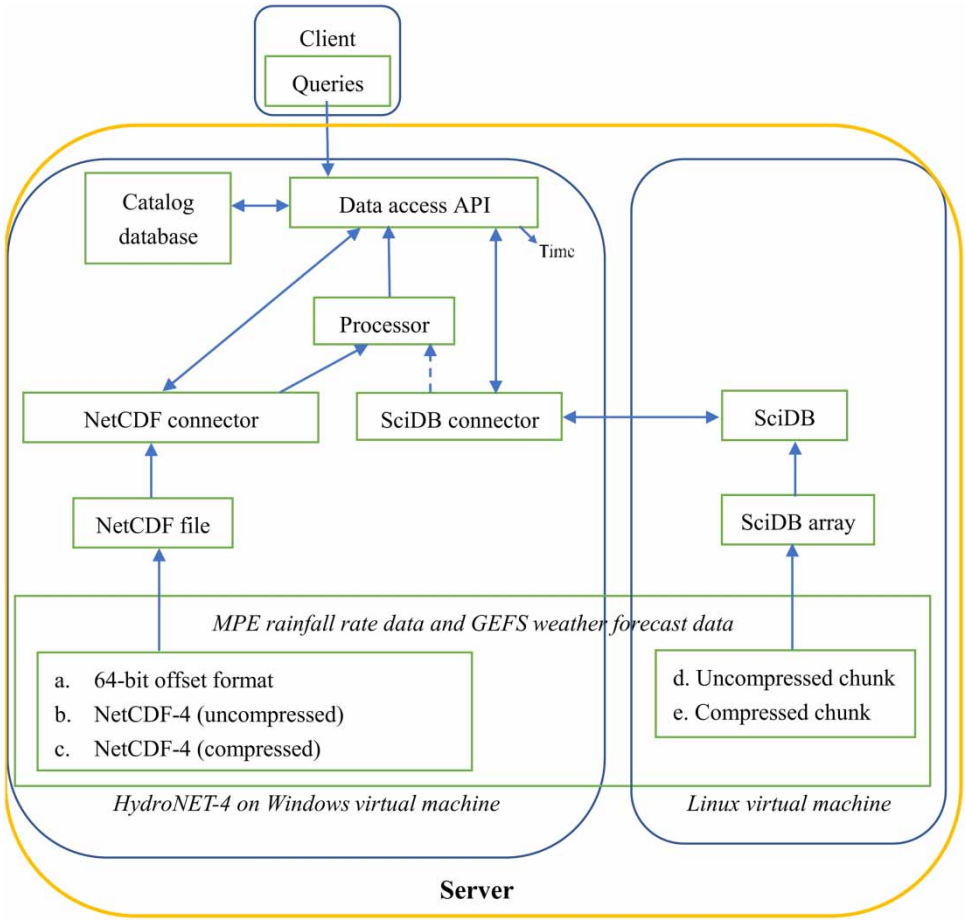

Figure 2 | Benchmark architecture.

amount of data are stored locally on internal servers of Hydrologic BV, it is convenient to transfer data through the internal network to reduce time cost. Another reason to adopt an internal server originates from the confidential aspect. This benchmark set-up is capable of tracing bottlenecks for big data management and thus indicates optimal solutions.

RESULTS AND DISCUSSION

Results of the MPE and GEFS data storage and query tests will be described individually in the following paragraphs. Data used for benchmarking are stored in NetCDF 64-bit offset format, NetCDF-4 format, and SciDB separately with different chunk sizes/dimensions and compression settings (Tables 5 and 6). Regarding benchmark testing, a query is executed 20 times alternating among different storage solutions. The final figure for the query response time is the average of the middle 12 records with the slowest four and the fastest four records removed for each solution. Benchmark performance is provided in Figures 3 and 4.

MPE

Two groups are established to investigate the performance of the time series extraction: medium size (1 day) and very large size (30 days). The first test is to extract time series of various lengths from NetCDF files and SciDB medium-size arrays (Figure 3). The query results indicate that the NetCDF-4 solution without compression and SciDB_100 solutions are the fastest. SciDB solutions with large chunk sizes take more time to execute the query. The negative

Table 5 | MPE storage of NetCDF solutions and SciDB (a solution name with '_C' represents that compression is used)

Solution name	Chunk size (longitude × latitude × time)	Chunk count	Average chunk size	Total storage size
NetCDF_64bit_offset_tiny	–			500 M
NetCDF4_4000_tiny	4,000 × 4,000 × 1	8	62.5 M	500 M
NetCDF4_4000_C_tiny	4,000 × 4,000 × 1	8	750 K	6 M
SciDB_4000_tiny	4,000 × 4,000 × 1	8	5 M	40.1 M
SciDB_4000_C_tiny	4,000 × 4,000 × 1	8	1.4 M	11.1 M
SciDB_800_tiny	800 × 800 × 1	200	0.2 M	40.2 M
SciDB_800_C_tiny	800 × 800 × 1	200	56.6 K	11.3 M
SciDB_100_tiny	100 × 100 × 1	12,800	3.3 K	42.3 M
SciDB_100_C_tiny	100 × 100 × 1	12,800	1 K	12.8 M
NetCDF_64bit_offset_small	–			1.46 G
NetCDF4_4000_small	4,000 × 4,000 × 1	24	60.8 M	1.46 G
NetCDF4_4000_C_small	4,000 × 4,000 × 1	24	750 K	18 M
SciDB_4000_small	4,000 × 4,000 × 1	24	4.8 M	115.9 M
SciDB_4000_C_small	4,000 × 4,000 × 1	24	1.3 M	32.1 M
SciDB_800_small	800 × 800 × 1	600	0.2 M	116.4 M
SciDB_800_C_small	800 × 800 × 1	600	55 K	32.8 M
SciDB_100_small	100 × 100 × 1	38,400	3.2 K	122.2 M
SciDB_100_C_small	100 × 100 × 1	38,400	1 K	37.1 M
NetCDF_64bit_offset_medium	–			5.86 G
NetCDF4_4000_medium	4,000 × 4,000 × 1	96	61 M	5.86 G
NetCDF4_4000_C_medium	4,000 × 4,000 × 1	96	750 K	72 M
SciDB_4000_medium	4,000 × 4,000 × 1	96	5.1 M	489 M
SciDB_4000_C_medium	4,000 × 4,000 × 1	96	1.4 M	136.2 M
SciDB_800_medium	800 × 800 × 1	2,400	200 K	491 M
SciDB_800_C_medium	800 × 800 × 1	2,400	58 K	139 M
SciDB_100_medium	100 × 100 × 1	153,600	3.4 K	514.7 M
SciDB_100_C_medium	100 × 100 × 1	153,600	1 K	157.3 M
NetCDF_64bit_offset_large	–			41 G
NetCDF4_4000_large	4,000 × 4,000 × 1	672	61 M	41 G
NetCDF4_4000_C_large	4,000 × 4,000 × 1	672	750 K	504 M
SciDB_800_large	800 × 800 × 1	16,800	180 K	2.98 G
SciDB_800_C_large	800 × 800 × 1	16,800	51 K	864 M
NetCDF_64bit_offset_vlarge	–			176 G
NetCDF4_4000_vlarge	4,000 × 4,000 × 1	2,880	61.1 M	176 G
NetCDF4_4000_C_vlarge	4,000 × 4,000 × 1	2,880	730 K	2.1 G
SciDB_800_vlarge	800 × 800 × 1	72,000	200 K	13.7 G
SciDB_800_C_vlarge	800 × 800 × 1	72,000	58 K	3.88 G

effect of compression on SciDB arrays with chunk size 4,000 × 4,000 × 1 is significant. However, when the chunk size is small, compression does not have a negative impact. The DEFLATE compression of NetCDF-4 causes severe degradation of the query performance. As the amount of data increase, the average time to extract a time

Table 6 | GEFS storage of NetCDF solutions and SciDB

Array name	Dimension order	Chunk size	Chunk count	Average chunk storage size	Total storage size
NetCDF_64bit_offset	–	–	–	–	1.55 G
NetCDF4_S3	X, Y, Forecast, Ensemble, Modelrun	360 × 181 × 1 × 20 × 1	40	40 M	1.55 G
NetCDF4_S3_C	X, Y, Forecast, Ensemble, Modelrun	360 × 181 × 1 × 20 × 1	40	16.4 M	654 M
NetCDF4_S5	X, Y, Forecast, Ensemble, Modelrun	360 × 181 × 1 × 1 × 1	800	2 MB	1.55 G
NetCDF4_S5_C	X, Y, Forecast, Ensemble, Modelrun	360 × 181 × 1 × 1 × 1	800	700 K	561 M
SciDB_S1	Modelrun, Ensemble, Forecast, Y, X	1 × 20 × 1 × 181 × 360	40	6.7 M	268 M
SciDB_S1_C	Modelrun, Ensemble, Forecast, Y, X	1 × 20 × 1 × 181 × 360	40	2.2 M	86.4 M
SciDB_S2	Modelrun, Forecast, Y, X, Ensemble	1 × 1 × 181 × 360 × 20	40	6.2 M	247 M
SciDB_S2_C	Modelrun, Forecast, Y, X, Ensemble	1 × 1 × 181 × 360 × 20	40	2 M	79 M
SciDB_S3	X, Y, Forecast, Ensemble, Modelrun	360 × 181 × 1 × 20 × 1	40	6.2 M	246.5 M
SciDB_S4	Ensemble, X, Y, Forecast, Modelrun	20 × 360 × 181 × 1 × 1	40	5.8 M	232.4 M
SciDB_LS1	Modelrun, Ensemble, Forecast, Y, X	1 × 20 × 40 × 181 × 360	1	268 M	268 M
SciDB_LS2	Modelrun, Forecast, Y, X, Ensemble	1 × 40 × 181 × 360 × 20	1	246.7 M	246.7 M
SciDB_LS3	X, Y, Forecast, Ensemble, Modelrun	360 × 181 × 40 × 20 × 1	1	246.5 M	246.5 M
SciDB_LS4	Ensemble, X, Y, Forecast, Modelrun	20 × 360 × 181 × 40 × 1	1	250 M	250 M

series of one time step reduces for NetCDF-4, SciDB_800, and SciDB_100 solutions. The scalability of the other solutions remains constant.

The second test compares the 64-bit offset, NetCDF4_4000, SciDB_800, and SciDB_800_C solution for extracting time series of 30 consecutive days of data. The uncompressed NetCDF-4 solution holds the leading position, while the 64-bit offset solution is nearly ten times slower than the NetCDF-4 solution. Further, the NetCDF-4 solution presents the pattern that the average time to extract the time series of a time step decreases when more data are requested. From raw test records, it is found that NetCDF-4's favorable scalability is due to the caching mechanism of the Windows operating system: after the first query execution, response time decreases dramatically to a certain threshold. The DEFLATE compression which is the default compression method on SciDB array does not have significant effect on the query performance. For SciDB solutions, an odd pattern arises that the average time to extract a time series of a time step reaches its minimum in the 96-step case (i.e., the time consumed to extract the time series divided by 96), while it rises again afterwards. The reason is that during the whole process of executing the query 20 times, relevant data are cached into the memory gradually instead of immediate caching after executing the same query two or three times.

Apart from the time series extraction, the results of the sub grid selection show that, in general, NetCDF-4 solution

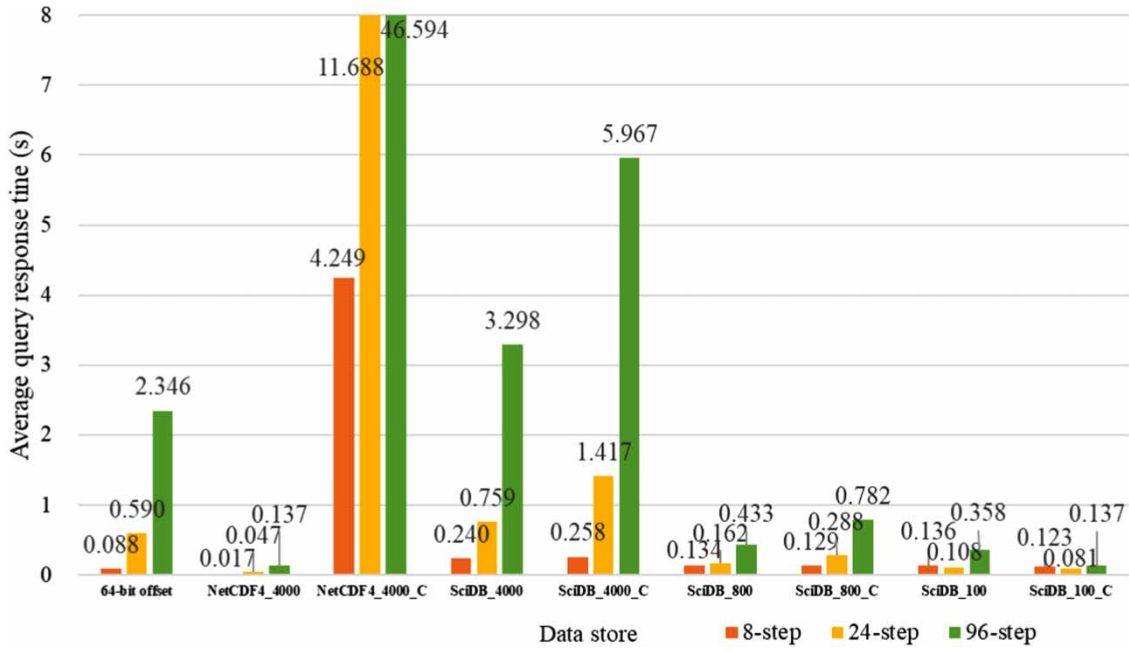

Figure 3 | The performance on retrieving MPE time series of different lengths, i.e., 8, 24, and 96 steps from a single location in the Indian Ocean at the medium-size level.

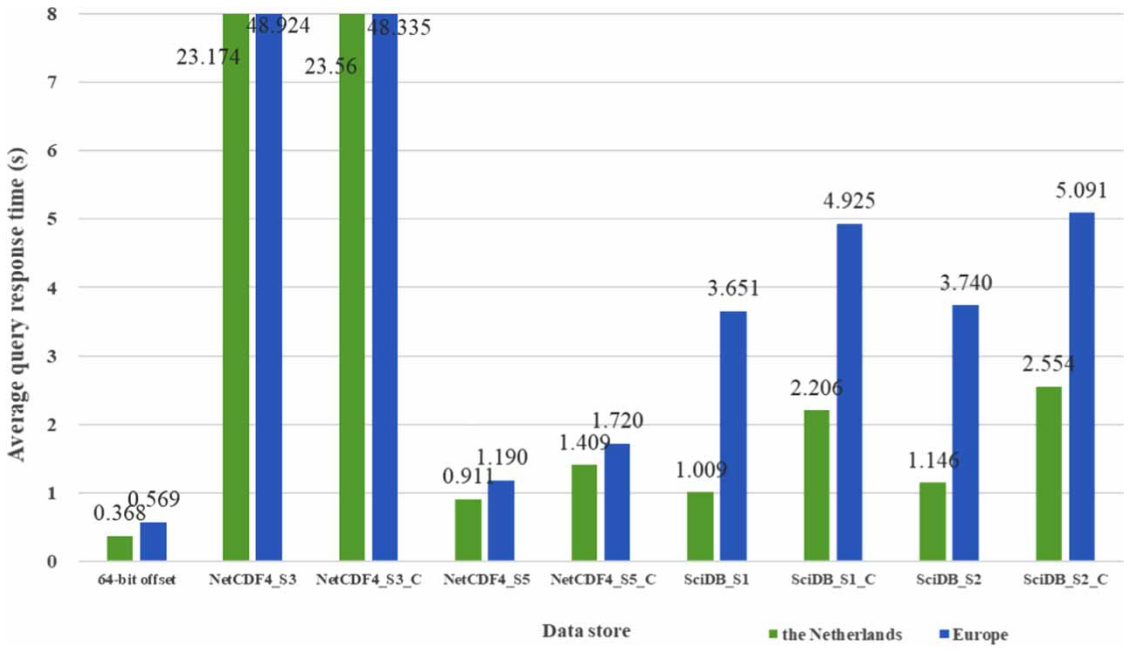

Figure 4 | The performance of calculating the ensemble mean of GEFS total precipitation in the Netherlands and Europe.

without compression is faster than the other solutions. The NetCDF 64-bit offset solution ranks the second, followed by other SciDB solutions. However, the gap is not significant for the SciDB solutions with small chunk sizes including $800 \times 800 \times 1$ and $100 \times 100 \times 1$. When the DEFLATE compression is applied to the NetCDF-4 storage, the query performance drops drastically and it takes around 40 times longer to extract the sub-grid than

extraction from uncompressed NetCDF-4 files. SciDB originally builds an index on Run Length Encoding (RLE) data. The DEFLATE compression, i.e., a secondary compression, does not cause much delay. All solutions scale well and performance remains at a constant level as the data size gets larger.

With respect to the average calculation, the 64-bit offset, NetCDF4_4000, SciDB_800, and SciDB_800_C solution are compared. NetCDF-4 solution without compression still leads the ranking. The performance of the different SciDB solutions does not vary much. The combination of the operator 'aggregate' and 'between' for the average calculation inside SciDB results in much more overhead than the case where only the 'between' operation is utilized for sub-selection. Inefficient processing ability with combined operators is a problem indicated by SciDB developers. A noticeable point for the NetCDF-4 solution is that when the average is calculated for 2,880 time steps, the average query response time suddenly increases dramatically. Raw measurements show that the gaps among all 20 measurements are fairly small. This indicates that the Windows operating system cached little relevant data. The reason is that after one query execution of the NetCDF-4 solution, another process requesting the 64-bit offset data follows. Hence, the query executed for the 64-bit offset solution flushes NetCDF-4 data cached. Average calculation with fewer time steps results in smaller output sizes, which is probably the reason why the cache flushing does not occur, as data still fit in the memory.

GEFS

In Table 6, the last dimension in the second column is the outermost dimension, which is the most significant in organizing the storage. S1, S2, S3, and S4 represent four data schemes, by sorting cells according to different order of dimensions. For instance, in a chunk of the SciDB_S3, data stored first are 360 cells of all X values where the Y and Ensemble value both equal 1. What are stored next are another 360 cells with Y equal to 2 while Ensemble remains unchanged. The chunk sizes of these four arrays keep at the same modest level, i.e., 1 value in the model run and the forecast dimension, 20 values in the ensemble dimension, 181 values in the Y dimension, and 360 values in the X dimension. In the S5 scheme, a chunk only contains a spatial grid, which is a decision from recommendations of Lee *et al.* (2008).

It is commonly accepted that retrieving data which are stored adjacently on the disk should be faster than data stored discontinuously. Thus, considering the dimension order, the SciDB_S1 should take less time to execute Q1 than the SciDB_S2, while the SciDB_S4 should respond faster than the SciDB_S3. In addition, the SciDB_S1 and the SciDB_S4 should both respond more quickly than either the SciDB_S2 or the SciDB_S3. This is verified through the benchmark tests. To exclude the effect of chunk size on the performance, GEFS dataset is restructured into another four arrays with large chunk size, i.e., LS1, LS2, LS3, and LS4. In these cases, the impact of dimension order on the query performance becomes more significant. However, an exception occurs in that the SciDB_LS2, in which ensemble values in a single location at one forecast step are stored discontinuously, responds faster than the SciDB_LS4 where ensemble values are stored successively.

Regarding Q4 *Selection of the mean of 20 ensembles of total precipitation for 40 forecast steps in the Netherlands (a bounding box)*, to investigate the scalability of different solutions, two regions are selected (Figure 4). One is the Netherlands containing 20 cells (5×4) and the other is the whole of Europe covering 3,888 cells (72×54). For the two cases, the pattern of performance is analogous to results of Q2 and Q3. Thanks to caching, compression of the NetCDF4_S5 causes less negative impact on the query performance compared with SciDB solutions. NetCDF-4 solutions with larger chunk size, i.e., S3, do not benefit from caching which is undermined by query execution on other data stores. In general, the NetCDF4_S5 and NetCDF4_S5_C present the best scalability. As the query area increases 200%, the query response time only grows 20–30%. The query processing time of the 64-bit offset solution experiences a 54% increase. With little caching, NetCDF-4 S3 data stores cost twice the time on the European scale compared to the Netherlands scale. The inefficient combination of 'aggregate' and 'between' operator for the ensemble mean calculation still confines the scalability of SciDB solutions.

CONCLUSIONS

Through the establishment of benchmark sets, construction of benchmark environment and tests, the research indicates that the solution of chunked NetCDF-4 without compression and the 64-bit offset solution outperform SciDB solutions for spatial selections and aggregations. The NetCDF-4 solution shows significant superiority for temporal selections as well. The setting of dimensions inside a chunk effectively gives the users control over how to cluster the MD spatio-temporal data. Depending on the data, the compression is typically in the range between a factor 3 (GFES data) up to nearly a factor 100 (MPE data). Although applications are confined in the realm of hydrology, the knowledge acquired can be applied to all MD array data management from the following aspects:

1. The chunk size plays an important role in the query performance and small chunk sizes are preferable for solutions adopting chunked storage. However, as is indicated in the research, indexing large quantities of small chunks could cause overload of the main memory.
2. The compression techniques, such as RLE and DEFLAT, can dramatically reduce the data storage. However, unless an indexing approach is developed to locate specific cells, compressed data may not be directly used for efficient querying.
3. The caching at the operating system or the database level can be complex during query execution. Cached data of one query may be flushed by another query executed which returns a large volume of data. Also, for some solutions, caching is too slow to present its advantages, which is also related to query types.
4. Changing the order of dimensions inside each chunk can have an influence on the query performance. However, when chunk size reduces to a modest level, the impact of dimension order decreases as well. It thus highlights the importance of an optimal chunk size and a smart indexing strategy from another perspective.

From this research, geo-data suppliers could start distributing various data products using state-of-the-art formats including NetCDF-4 and HDF-5. Both share the same chunked storage model (Pourmal 2007). Other formats like GRIB and GRD are conventionally efficient for storage, but they bring much additional processing work for end users. Another option for data publishers would be storage and processing services through either an integrated or distributed MD array DBMS which provides advantageous integration with different data types, smart caching strategies, transaction support, and out-of-the-box support for parallelization. While unifying data models and formats might still be a long-term goal for which to strive, practical suggestions on managing MD data for geo-related communities are listed below:

1. Before importing all data into specific DBMSs, modelers may first spend some time on improving available file-based solutions and try to update from contiguous storage to a chunked storage approach. Slow data loading is a common problem and thus a bottleneck for database solutions, so this should receive attention by the developers.
2. For a given storage solution, it is recommended to optimize the chunk size for the most important queries. If a sufficiently fast solution for all queries cannot be found, a multiple representation solution can be considered. In the research, MPE chunks tested are of a 2D shape, that is, chunk size on the temporal dimension is equal to 1. In fact, cubic chunks can be another choice. More benchmark tests should be performed after modifying the initial chunk size to achieve the best performance.
3. An advanced platform is a general way to improve the efficiency of data management. For example, parallelization can facilitate big data loading, updating and query processing, especially with recently developed Hadoop/Spark platforms. As a file-based solution normally needs extra development for parallelization, DBMSs with native realization of functions, prior assessment of time and financial cost should be utilized.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the funding support from China National Key Research and Development Program (Grant No. 2017YFC0405801-02) and thank Hydrologic BV for the financial and technical support during this research.

REFERENCES

Abebe, A. J. & Price, R. K. 2005 Decision support system for urban flood management. *Journal of Hydroinformatics* **7** (1), 3–15.

Ashrit, R., Iyengar, G. R., Sankar, S., Ashish, A., Dube, A., Dutta, S. K., Prasad, V. S., Rajagopal, E. N. & Basu, S. 2013 *Performance of Global Ensemble Forecast System (GEFS) During Monsoon 2012*. NCMRWF Research report, NMRF/RR/1. http://www.ncmrwf.gov.in/GEFS_Report_Final.pdf (accessed 14 December 2017).

Baumann, P., Dehmel, A., Furtado, P., Ritsch, R. & Widmann, N. 1998 The multidimensional database system RasDaMan. In: *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, USA, pp. 575–577.

Brown, P. G. 2010 Overview of SciDB: large scale array storage, processing and analysis. In: *Proceedings of the 2010 International ACM SIGMOD Conference on Management of Data*, Indianapolis, Indiana, USA, pp. 963–968.

Cohen, S., Hurley, P., Schulz, K. W., Barth, W. L. & Benton, B. 2006 Scientific formats for object-relational database systems: a study of suitability and performance. *ACM SIGMOD Record* **35** (2), 10–15.

Colliat, G. 1996 OLAP, relational, and multidimensional database systems. *ACM SIGMOD Record* **25** (3), 64–69.

Crétaux, J. F., Jelinski, W., Calmant, S., Kouraev, A., Vuglinski, V., Bergé-Nguyen, M., Gennero, M. C., Nino, F., Abarco Del Rio, R., Cazenave, A. & Maisongrande, P. 2011 SOLS: A lake database to monitor in the Near Real Time water level and storage variations from remote sensing data. *Advances in Space Research* **47** (9), 1497–1507.

Cudre-Mauroux, P., Kimura, H., Lim, K. T., Rogers, J., Madden, S., Stonebraker, M., Zdonik, S. B. & Brown, P. 2012 SS-DB: A standard science DBMS benchmark. http://www-conf.slac.stanford.edu/xldb10/docs/ssdb_benchmark.pdf (accessed 14 December 2017).

Gonçalves, R., Zlatanova, S., Kyzirakos, K., Nourian, P., Alvanaki, F. & van Hage, W. 2016 A columnar architecture for modern risk management system. In: *Proceedings of the IEEE 12th International Conference on E-Science*, Baltimore, Maryland, USA, pp. 424–429.

Heinemann, T., Latanzio, A. & Roveda, F. 2002 The Eumetsat multi-sensor precipitation estimate (MPE). In: *Second International Precipitation Working Group (IPWG) Meeting*, Madrid, Spain, pp. 23–27.

Idreos, S., Groffen, F., Nes, N., Manegold, S., Mullender, K. S. & Kersten, M. 2012 MonetDB: two decades of research in column-oriented database architectures. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* **35** (1), 40–45.

InterSystems. 2017 Using Caché Globals. http://docs.intersystems.com/documentation/cache/20172/pdfs/GGBL.pdf (accessed 14 December 2017).

Lee, C., Yang, M. & Aydt, R. 2008 *NetCDF-4 Performance Report*, Technical report, HDF group. https://support.hdfgroup.org/pubs/papers/2008-06_netcdf4_perf_report.pdf (accessed 14 December 2017).

Liu, H., van Oosterom, P., Hu, C. & Wang, W. 2016 Managing large multidimensional array hydrologic datasets: a case study comparing NetCDF and SciDB. *Procedia Engineering* **154**, 207–214.

Oracle 2008 *Oracle Essbase Database Administrator's Guide*. http://docs.oracle.com/cd/E12825_01/epm.111/esb_dbag/frameset.htm?dinconc.htm (accessed 14 December 2017).

Oracle 2014 *Spatial and Graph GeoRaster Developer's Guide*. http://docs.oracle.com/database/121/GEORS/toc.htm (accessed 14 December 2017).

Pedersen, T. B. & Jensen, C. S. 2001 Multidimensional database technology. *Computer* **34** (12), 40–46.

Pourmal, E. 2007 *What NetCDF Users Should Know About HDF5?* https://www.unidata.ucar.edu/software/netcdf/workshops/2007/hdf5/ncw07-hdf5.pdf (accessed 14 December 2017).

Reichard, L., Lobbrecht, A., Clark, S., Catalano, C., Tate, B. & Cox, D. 2014 Supporting water managers making effective decisions by using HydroNET. In: *Proceedings of the 35th Hydrology and Water Resources Symposium*, Perth, Australia, pp. 710–717.

Rew, R., Hartnett, E. & Caron, J. 2006 NetCDF-4: Software implementing an enhanced data model for the geosciences. In: *Proceedings of the 22nd AMS Conference on Interactive Information and Processing Systems for Meteorology*, Atlanta, Georgia, USA. AMS Press, pp. 6.6.

Stonebraker, M., Brown, P., Poliakov, A. & Raman, S. 2011 The architecture of SciDB. In: *Proceedings of the 23rd International Conference on Scientific and Statistical Database Management*, Portland, Oregon, USA, pp. 1–16.

Su, Y. & Agrawal, G. 2012 Supporting user-defined subsetting and aggregation over parallel netcdf datasets. In: *Proceedings of the International Symposium on Cluster, Cloud and Grid Computing*, Ottawa, Canada. IEEE Press, pp. 212–219.

Trepanier, M., Gauthier, V., Besner, M. C. & Prevost, M. 2006 A GIS-based tool for distribution system data integration and analysis. *Journal of Hydroinformatics* **8** (1), 13–24.

Xie, Q. J. 2016 The design of a high performance earth imagery and raster data management and processing platform. In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, Prague, Czech Republic, pp. 551–555.

First received 16 October 2017; accepted in revised form 17 April 2018. Available online 10 May 2018