Managing Large Multidimensional Array Hydrologic Datasets
A Case Study Comparing NetCDF and SciDB

Liu, H.; van Oosterom, P.J.M.; Hu, C.; Wang, Wen

DOI
10.1016/j.proeng.2016.07.449

Publication date
2016

Document Version
Final published version

Published in
Procedia Engineering

Citation (APA)
Liu, H., van Oosterom, P. J. M., Hu, C., & Wang, W. (2016). Managing Large Multidimensional Array Hydrologic Datasets: A Case Study Comparing NetCDF and SciDB. *Procedia Engineering*, *154*, 207-214. https://doi.org/10.1016/j.proeng.2016.07.449

Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 154 (2016) 207 – 214

Procedia Engineering

www.elsevier.com/locate/procedia

12th International Conference on Hydroinformatics, HIC 2016

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

Haicheng Liu[a,*], Peter van Oosterom[b], Chengfang Hu[a], Wen Wang[c]

[a] *Section Digital Basins, Department Spatial Information, CRSRI, No.289 HuangPu Road, 430010, Wuhan City, China*
[b] *Section GIS technology, Department OTB, Faculty of Architecture and the Built Environment, TU Delft, Julianalaan 134, 2628 BL Delft, The Netherlands*
[c] *State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering, Hohai University, Nanjing, 210098, China*

Abstract

Management of large hydrologic datasets including storage, structuring, indexing and query is one of the crucial challenges in the era of big data. This research originates from a specific data query problem: time series extraction at specific locations takes a long time when a large multidimensional dataset is stored in non-chunked NetCDF classic or 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 multidimensional (MD) array database management system (DBMS) applying chunked storage structure are benchmarked to determine the best solution for storing and querying large hydrologic datasets. To achieve this, expert consultancy was conducted to establish benchmark sets. To guarantee a fair benchmark test environment, HydroNET-4 system was utilized and adapters for NetCDF files and SciDB were developed to manage and query data. In final benchmark tests, effect of data storage configurations such as chunk size and compression on query performance is also explored. Results indicate that SciDB arrays utilizing small chunk sizes show favorable performance. However with current implementation of SciDB, large numbers of small chunks cause huge overload of main memory which constraints SciDB's scalability. Compression of SciDB can either have negative or no effect on query performance, while it causes significant query degradation to NetCDF-4 solution. The research illustrates that for big hydrologic array data management, the properly chunked NetCDF-4 solution without compression is in general more efficient than the SciDB DBMS. So under current big data environment, traditionally adopted file-based hydroinformatic solutions can still be applicable after proper updating.

Keywords: benchmark; NetCDF; SciDB; chunked storage structure; hydrologic dataset;

* Corresponding author. Tel.: 008615902732003; fax: 008602782820076. *E-mail address:* liuhaicheng@mail.crsri.cn

doi:10.1016/j.proeng.2016.07.449

1. Introduction

In hydrologic domain, original data collecting techniques with improved accuracy become increasingly prevalent currently, radar systems for instance. Meanwhile, new sensor platforms and sources such as citizen-supplied observations are arising all the time. All kinds of simulation models never stop running to produce essential results for decision making. These large amounts of datasets are normally stored and distributed in diverse formats, which brings inconvenience for professionals to extract effective information for certain applications efficiently. These commonly used formats include Hierarchical Data Format (HDF), Network Common Data Form (NetCDF) and Gridded Binary (GRIB) which are originally designed for meteorological purposes. Among them, NetCDF is notable for its simple data model, ease of use, portability, and strong user support infrastructure [1]. It is widely applied to record meteorological, oceanographic as well as hydrologic observation or simulation results.
However, according to practical experience, traditional NetCDF solutions perform inefficiently in retrieving information from large spatio-temporal datasets for certain queries. This is caused by the way it stores variable values, which is known as contiguous storage structure. Basically, for a grid full of variable values in a certain spatial area, NetCDF stores values into a one-dimensional array according to a row-majored order (Fig.1. a-c). And in this way, to query the value in a particular cell, calculating position of the cell in the one-dimensional array is needed. Extraction of a time series (Fig.1. a) becomes more expensive due to accessing required cell values, which are widely spread over the disk in a one-dimensional array (Fig.1. c). Alternatively, it is possible to store time series of every location as a one-dimensional variable in NetCDF, but then retrieving the complete grid at a single time step becomes the problem. Although chunked storage structure is introduced to the later version of NetCDF, i.e. NetCDF-4, contiguous storage structure is still the solution applied in most cases as we learnt through discussions with geo-data experts as well as checking NetCDF data offered by popular providers on the Internet.

When managing large numbers of MD array datasets, it is natural to adopt a DBMS solution as it could provide an easier to use and more scalable alternative. In practice, organizations can have a range of different datasets and types: administrative data, point cloud data, temporal data, etc. A standardized and generic DBMS solution would be preferable when combining, for example, vector and raster data in a query. In addition, NetCDF file-based solutions normally offer a limited range of functionalities while DBMS have rich functionality thanks to ad hoc query support and declarative programming models. Most modern DBMSs also offer automatic parallelization in query execution. Within DBMS scope, MD array DBMS is optimized further for MD array data management. It can specify metadata [2] and supports storage of MD arrays. It employs the chunked storage structure (Fig.1. d-e) which divides a whole dataset into separate chunks with specified chunk sizes [3,4]. Based on this storage structure, MD array DBMSs then apply specific array addressing and relative offset calculation to index values, which is proved to be of high query efficiency [5]. Hence, this research is aimed at investigating whether the MD array DBMS can achieve better performance in processing queries on large MD hydrologic datasets than classic non-chunked NetCDF solution and competitive performance when compared to chunked NetCDF-4 file based solution.

To address the main research question, benchmark tests are executed. Several steps should be performed: define benchmark, select MD array DBMS, create benchmark test environment, execute benchmarks and analyze results. Several solutions are going to 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

The structure of this paper is as follows: In Section 2, expert consultancy to determine final queries and datasets is described. This is then followed by an explanation of the benchmark test environment and discussion of benchmark results in Section 3. The paper finishes with conclusions and future work.

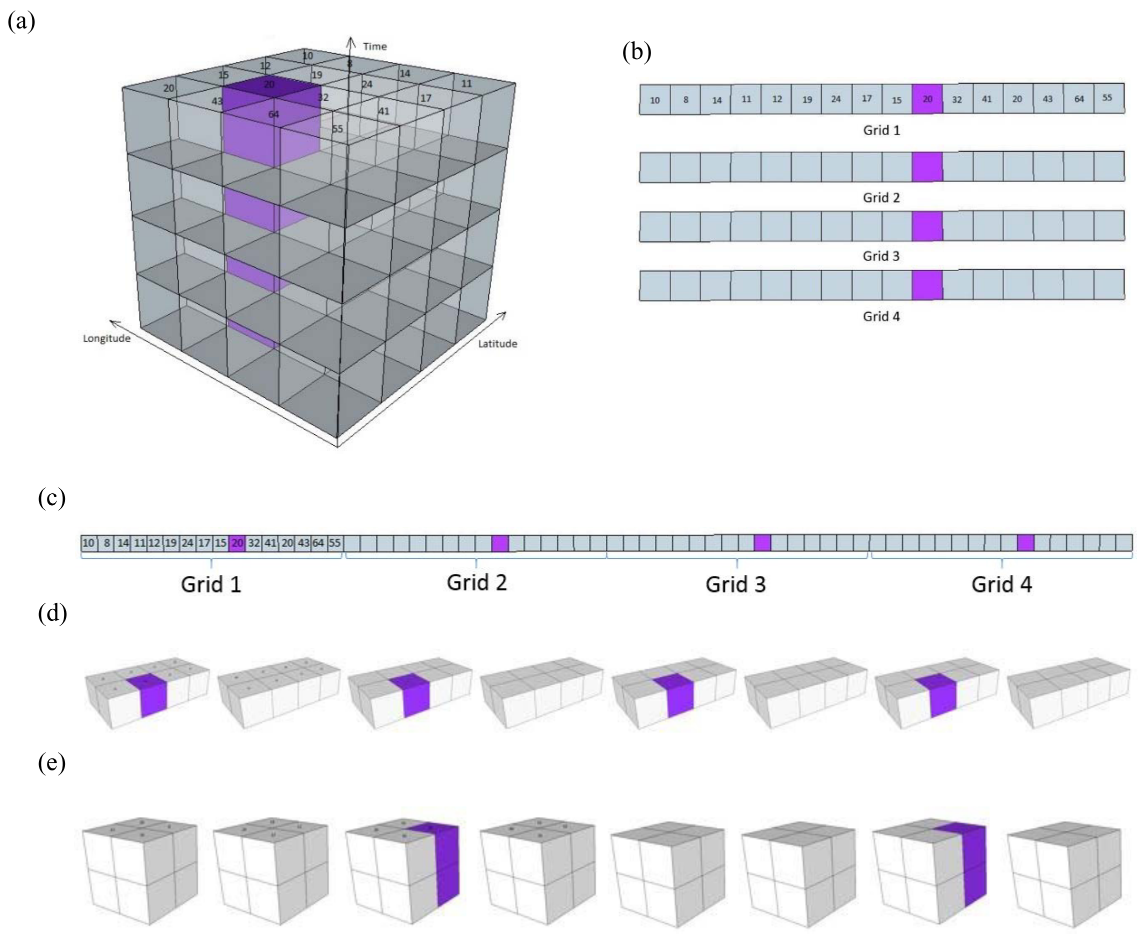


Fig.1. storing a three dimensional precipitation dataset with contiguous storage structure and chunked storage structure. (a) a sample 3 dimensional precipitation dataset and a time series at a specific location shown in purple. Only cell values in the first grid are shown; (b) storing each grid in a row-majored order. Purple cells are in the time series; (c) concatenating all grids into a one-dimensional array, which is the contiguous storage structure (d) storage of the precipitation dataset with chunked storage structure while the chunk size is 2 (longitude) x 2 (latitude) x 2 (time). Each chunk is also stored as one dimensional array on disk in the end but chunks are independent from each other; (e) chunked storage of the precipitation dataset with 2 (longitude) x 4 (latitude) x 1 (time) as chunk size.

2. Preparation for benchmark test

In order to assess the various solutions for managing MD array data it is important to clearly specify representative types of data and corresponding series of queries. Performance of different solutions depends on how similar the data is organized and stored compared to the requested selection. Designing 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 were first interviewed. Next a representative dataset and set of queries were specified for the benchmark.

In total 6 hydrologic experts are consulted. In this research, processes on data that can be executed with SQL are regarded as queries based on selection and operations (sum, difference, min, max, avg,..). Tasks such as numerical computations and simulations provided by professional software are not included. To clearly define query types a classification is performed (Table 1).

Table 1. Classes of queries collected from consultancy

Query/operation class	
A. Selection based on dimension value	B. Selection based on variable value
C. Spatial join/ combination/ masking operations	D. Mathematical calculation (sum, avg,…)

Based on previous related studies [6,7] and practical experience, four main query classes are defined. In most cases, users first select the area of interest or a certain period of time (Class A), i.e. selection according to spatio-temporal dimension values and then focus on variable values. Class D, i.e. mathematic calculation is also quite often applied by hydrologists. While class B is the least crucial type. As to spatial operations such as joining or combining spatial data (class C), this is not very common in the daily work of hydrologic experts. Therefore, classes A and D get priority in the benchmark. In practice, classes are often combined as one query type.

The size of datasets for benchmarking needs to be sufficiently large and has the potential to exceed TB level. Dimensions of the dataset should focus on spatial and temporal dimensions, this conforms to the datasets described by experts. After considering all requirements for datasets, i.e. data size, dimension and accessibility, the MPE (Multi-Sensor Precipitation Estimate) dataset is selected for testing (Table 2).

Table 2. MPE dataset for benchmarking

Information stored	Dimension count	Temporal resolution	Spatial resolution and coverage	Dimension Span (single file)	Single file size	Data format
Rainfall rate (IRRATE); Availability; Quality	3	15 minutes	0.03 degree (3.3 km), 1/3 world	x, y, time (4 000, 4 000, 4)	250 MB	64-bit offset

Dataset MPE stores the rainfall rate data processed from raw satellite data. The Availability information indicating whether a grid at a certain time step is missing and the Quality marking if the satellite data have been corrected according to ground measurements are also recorded. Hydrologic Research BV can provide records for more than two years, which results in the total amount of data larger than 4.18 TB (2 x 365 x 24 x 250 MB).

According to most important classes of query types and the dataset selected, conceptual queries for benchmarking are designed as follows:

- *Q1: Selection based on spatial dimensions for Delft (Class A)*
- *Q2: Selection based on spatial dimensions for north Netherlands (Class A)*
- *Q3: Extraction of time series for a single location in the Indian Ocean (Class A)*
- *Q4: Historical one month average value for the Netherlands (Class D)*

3. Result of benchmark test

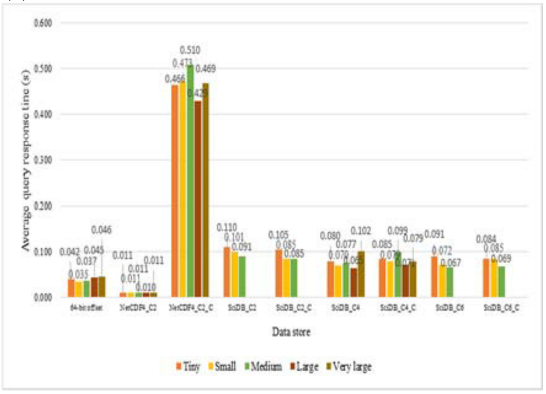
The benchmark test environment is based on the HydroNET-4 system [8] from Hydrologic Research B.V. The system can parse queries encapsulated in HTTP POST requests and then execute NetCDF queries using HydroNetCDF library developed and optimized by the company. While NetCDF-4 component is constructed on standard library NetCDF-4.1.3. SciDB database is communicated through a connecter developed on top of shim which is a super-basic SciDB client that exposes limited SciDB functionality through a simple HTTP API. The whole benchmark environment is installed on a Dell Inc. OptiPlex 745 server. The sever has one Intel processor with 2 cores, 6600 at 2.4 GHz, 4 x 2 GB DDR2 RAM, 3 TB SATA 5400 rpm Western Digital hard disk. Data used for benchmarking are stored in NetCDF 64-bit offset format, NetCDF-4 format and SciDB separately (Table 3). Regarding benchmarking, a query is executed 20 times alternating between different storage solutions. The final number for query response time is the average of the middle 12 time records with the largest 4 and the smallest 4 values removed for each solution. Benchmark performance for different solutions is provided in Fig. 2.

Table 3. Storage details of NetCDF solutions and SciDB

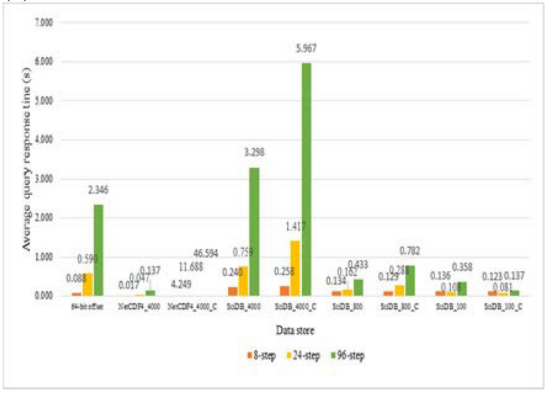
Solution name	Chunk size (Longitude x Latitude x Time)	Compression (y/n)	Total storage size
NetCDF_64bit_offset_tiny	-	n	500M
NetCDF4_4000_tiny	4 000 x 4 000 x 1	n	500M

NetCDF4_4000_C_tiny	4 000 x 4 000 x 1	y	6M
SciDB_4000_tiny	4 000 x 4 000 x 1	n	40.1M
SciDB_4000_C_tiny	4 000 x 4 000 x 1	y	11.1M
SciDB_800_tiny	800 x 800 x 1	n	40.2M
SciDB_800_C_tiny	800 x 800 x 1	y	11.3M
SciDB_100_tiny	100 x 100 x 1	n	42.3M
SciDB_100_C_tiny	100 x 100 x 1	y	12.8M
NetCDF_64bit_offset_small	-	n	1.46G
NetCDF4_4000_small	4 000 x 4 000 x 1	n	1.46G
NetCDF4_4000_C_small	4 000 x 4 000 x 1	y	18M
SciDB_4000_small	4 000 x 4 000 x 1	n	115.9M
SciDB_4000_C_small	4 000 x 4 000 x 1	y	32.1M
SciDB_800_small	800 x 800 x 1	n	116.4M
SciDB_800_C_small	800 x 800 x 1	y	32.8M
SciDB_100_small	100 x 100 x 1	n	122.2M
SciDB_100_C_small	100 x 100 x 1	y	37.1M
NetCDF_64bit_offset_medium	-	n	5.86G
NetCDF4_4000_medium	4 000 x 4 000 x 1	n	5.86G
NetCDF4_4000_C_medium	4 000 x 4 000 x 1	y	72M
SciDB_4000_medium	4 000 x 4 000 x 1	n	489M
SciDB_4000_C_medium	4 000 x 4 000 x 1	y	136.2M
SciDB_800_medium	800 x 800 x 1	n	491M
SciDB_800_C_medium	800 x 800 x 1	y	139M
SciDB_100_medium	100 x 100 x 1	n	514.7M
SciDB_100_C_medium	100 x 100 x 1	y	157.3M
NetCDF_64bit_offset_large	-	n	41G
NetCDF4_4000_large	4 000 x 4 000 x 1	n	41G
NetCDF4_4000_C_large	4 000 x 4 000 x 1	y	504M
SciDB_800_large	800 x 800 x 1	n	2.98G
SciDB_800_C_large	800 x 800 x 1	y	864M
NetCDF_64bit_offset_vlarge	-	n	176G
NetCDF4_4000_vlarge	4 000 x 4 000 x 1	n	176G
NetCDF4_4000_C_vlarge	4 000 x 4 000 x 1	y	2.1G
SciDB_800_vlarge	800 x 800 x 1	n	13.7G
SciDB_800_C_vlarge	800 x 800 x 1	y	3.88G

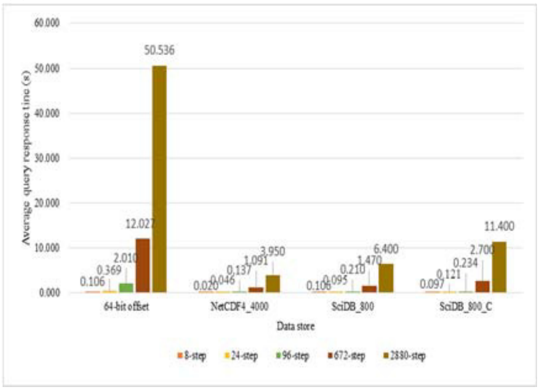
(a)



(b)



(c)



(d)

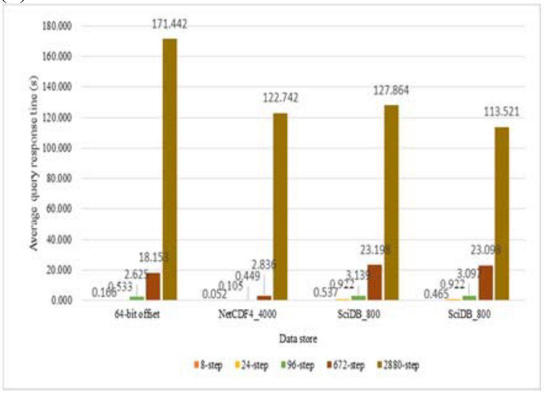


Fig. 2. performance of diverse solutions, (a) retrieving grid covering Delft at one time step; (b) retrieving time series of different lengths from a single location in the Indian Ocean at medium level; (c) retrieving time series of different lengths from a spot location in the Indian Ocean at very large level; (d) average calculation with different time steps at the Netherlands scale at very large level.

4. Discussions

A The results of the sub grid selection for Delft show that overall NetCDF-4 without compression is faster than other solutions (Fig.2. a). NetCDF 64-bit offset solution ranks second, followed by various SciDB solutions. However, by subtracting the additional noise introduced by SciDB connector, SciDB solutions can probably achieve the same performance as 64-bit offset solution, especially the arrays with small chunk sizes. When DEFLATE compression is applied to NetCDF-4 storage, the query performance degrades severely and it takes around 40 times longer to extract the sub grid than for uncompressed NetCDF-4 files. The compression of SciDB does not have significant influence on query response time. SciDB builds an index on RLE encoded data and the compression done by DEFLATE in this case, i.e. a secondary (de)compression, does not take that much time. All solutions scale well, that is performance keeps at a constant level as data size gets larger. Query performance on north Netherlands is not shown for the sake of its similar patterns to that of Delft.

Two test groups are developed to investigate performance of time series extraction: medium (1 day) and very large (30 day) datasets (Fig.2. b ~ c). The first test is to extract time series of various lengths from NetCDF files and SciDB medium arrays (1 day of data). Benchmark 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. The negative effect of compression on SciDB arrays with chunk size 4 000 x 4 000 x 1 is significant. For small chunk sizes, compression does not have a negative impact. The DEFLATE compression of NetCDF-4 on the other hand still causes severe degradation of query performance (Due to visual effect, only numbers representing query response time are shown in Fig.2. c). Regarding scalability, as the amount of data increases, the average time to extract a time series reduces for NetCDF-4, SciDB_800 and SciDB_100 solutions. For others, the scalability keeps at a constant level. The second test focuses on very large arrays of SciDB. Due to the large query response time of compressed NetCDF-4 solution in the initial tests, it is excluded from benchmarking. Uncompressed NetCDF-4 remains the fastest solution, the 64-bit offset solution is nearly 10 times slower than NetCDF-4. Besides, the NetCDF-4 solution once again presents the pattern that the average time to extract the time series of one time step decreases when more data is queried. From the raw test records, it is found that NetCDF-4's favorable scalability is due to the Windows caching mechanism: after the first query execution, response time decreases dramatically to a certain level. DEFLATE compression on SciDB array does not have significant effect on query performance. For SciDB solutions, an odd pattern arises that the average time to extract time series of one time step reach its minimum in the 96 steps case, while it rises again after that. The reason is that during the whole process of executing the query 20 times on a SciDB array, relevant data is cached into memory very slowly instead of cached completely into memory after executing the same query 2 or 3 times. So by removing the 4 slowest response time records, some slower response times will remain and be used for calculating average query response time.

The fourth query is average calculation (Fig.2. d). NetCDF-4 shows the best performance. The 64-bit offset solution comes second while SciDB solutions take more time to finish. Performance of the normal SciDB array and its compressed version do not vary much. The execution of average calculation is composed of two phases for NetCDF file solutions. First phase is to extract the related spatiotemporal sub cube from the whole dataset. Then aggregation is performed on the selected data later in a second phase. HydroNET-4 records required time for both phases, this indicates that average calculation is hundreds of times faster than sub selection. But the combination of operators “aggregate” and “between” for average calculation in SciDB results in much more overhead than only sub selection with the “between” operator. Inefficient processing ability with combined operators is a problem indicated by the SciDB team. A noticeable point for NetCDF solutions is that when the average is calculated for 2 880 time steps, the average query response time suddenly increases several times, especially for the NetCDF-4 solution. The raw measurements of the NetCDF-4 solution show that all 20 measurements are at nearly the same level. This indicates that Windows cached little relevant data for the aggregation query. The reason for this is that the query executed for the 64-bit offset solution flushes cached NetCDF-4 data due to the benchmarking method (i.e. For (int i = 0; i < 20; i++){Q4(64bitoffset); Q4(NetCDF-4); Q4(SciDB);}). Average calculation with less time steps results in smaller query results of sub selection, which is probably the reason why cache flushing does not occur in that case.

5. Summary

Queries and datasets frequently used by water experts are collected by means of interviews. After query classification and designing, specific datasets and queries used for benchmarking are determined. In total 9 criteria are defined to compare MD array DBMSs, as a result SciDB is selected for benchmarking. After constructing a test environment in the HydroNET-4 system, NetCDF and SciDB solutions are benchmarked and analyzed. The overall result indicates that the solution of NetCDF-4 without compression and 64-bit offset solution outperform SciDB solutions. Although SciDB query response time include additional noise like HTTP communication and data transfer, the noise does not change the final conclusion due to the large gap of overall query performance between solutions. As the benchmark results indicate, chunk size plays an important role in query performance and small chunk sizes are preferable for SciDB data management. But small chunk sizes in the current implementation are not feasible because it requires too much main memory. DEFLATE compression of SciDB can either have negative effect or no effect on query performance. DEFLATE compression of NetCDF-4, on the other hand causes significant query performance degradation. The benchmark tests also reveal the complexity of query execution. Cached NetCDF data may be flushed by another query executed and SciDB’s slow caching method makes it miss benefits as a result of caching. Nonetheless, based on the research results, a general suggestion for data management is that before importing all data into specific DBMSs, ICT architects can first spend some time on improving available file based solutions. Besides, for a given storage solution, it is suggested to optimize 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.

Acknowledgements

The authors would like to thank Tom Commandeur from Hydrologic Research B.V. and Theo Tijssen from TU Delft for the technical support during this research. Also financial supports from Hydrologic Research B.V. , the Specialized Research Fund for the Doctoral Program of Higher Education (20130094110007), the 111 Project (B08048) and National Natural Science Foundation of China (Grant No.41301435) are gratefully acknowledged.

References

[1] R. Rew, E. Hartnett, J. Caron, NetCDF-4: Software implementing an enhanced data model for the geosciences. In: 22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanograph and Hydrology, Atlanta, Georgia, USA, 2006.

[2] T. B. Pedersen, C. S. Jensen, Multidimensional database technology. Computer, 34(12) (2001), 40-46

[3] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, N. Widmann, The multidimensional database system RasDaMan. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, ACM SIGMOD Record Vol. 27, No. 2, ACM Press, 1998, pp. 575-577.

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

[5] G. Colliat, OLAP, relational, and multidimensional database systems. ACM SIGMOD Record Vol. 25, No. 3, ACM Press, 1996, pp. 64-69.

[6] Y. Su, G. Agrawal, 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, 2012, pp. 212-219.

[7] S. Cohen, P. Hurley, K. W. Schulz, W. L. Barth, B. Benton, Scientific formats for object-relational database systems: a study of suitability and performance. ACM SIGMOD Record Vol. 35, No. 2, ACM Press, 2006, pp. 10-15.

[8] L. Reichard, A. Lobbrecht, S. Clark, C. Catalano, B. Tate, D. Cox, Supporting water managers making effective decisions by using HydroNET. In: Hydrology & Water Resources Symposium, Australia, 2014.