Comparing NetCDF and SciDB on managing and querying 5D hydrologic dataset

Haicheng Liu¹ and Xiao Xiao¹

¹Changjiang River Scientific Research Institute, Changjiang River Water Resource Commission, Wuhan, China
E-mail: liuhaicheng@mail.crsri.cn

Abstract. Efficiently extracting information from high dimensional hydro-meteorological modelling datasets requires smart solutions. Traditional methods are mostly based on files, which can be edited and accessed handily. But they have problems of efficiency due to contiguous storage structure. Others propose databases as an alternative for advantages such as native functionalities for manipulating multidi- mensional (MD) arrays, smart caching strategy and scalability. In this research, NetCDF file based solutions and the multidimensional array database management system (DBMS) SciDB applying chunked storage structure are benchmarked to determine the best solution for storing and querying 5D large hydrologic modelling dataset. The effect of data storage configurations including chunk size, dimension order and compression on query performance is explored. Results indicate that dimension order to organize storage of 5D data has significant influence on query performance if chunk size is very large. But the effect becomes insignificant when chunk size is properly set. Compression of SciDB mostly has negative influence on query performance. Caching is an advantage but may be influenced by execution of different query processes. On the whole, NetCDF solution without compression is in general more efficient than the SciDB DBMS.

1. Introduction
In hydrologic domain, all kinds of simulation models never stop running to produce essential results for decision making. These modelling datasets are normally featured by high dimensionality, large data size and diverse storage formats, which brings inconvenience for professionals to extract effective information for certain applications efficiently. One of the most frequently used format is NetCDF which is notable for its simple data model, ease of use, portability, and strong user support infrastructure [1]. 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-major ordered. 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 becomes more expensive due to accessing required cell values, which are widely spread over the disk in a one-dimensional array. 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.

Chunked storage structure is introduced to the later version of NetCDF, i.e. NetCDF-4. The chunked storage structure divides a whole dataset into separate chunks with specified chunk sizes [2,3]. Based on this storage structure, specific array addressing and relative offset calculation are then applied to index values, which is proved to be of high query efficiency [4]. Within DBMS scope, MD array DBMS
also employs chunked storage structure. Additionally, it is optimized further for MD array data management such as plenty native functionalities to manipulate arrays, smart caching algorithms, etc [2]. It also supports efficient storage and query features like versioning and overlapping [4,5].

This research investigates whether the MD array DBMS SciDB can achieve better performance in processing queries on high dimensional hydrologic simulation datasets than classic non-chunked NetCDF solution, and if SciDB has 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, create benchmark test environment, execute benchmarks and analyse results.

2. Data and query

Queries and datasets frequently used by water experts should be collected for devising a scientific benchmark. Therefore, several experts were first interviewed. Next a representative dataset and set of queries were specified for the benchmark.

Global Ensemble Forecast System (GEFS) dataset (table 1; figure 1) is calculated from a global forecast model. It contains 5 dimensions. The forecast dimension refers to the time steps simulated while model run represents the time to run the forecast model in reality. Increments for both dimensions are 6 hours, i.e. every 6 hours the model will be run and forecast temporal interval is also 6 hours. Ensemble represents initial condition and 20 ensembles simulate 20 different initial conditions as the input for the forecast model. Only one sample is applied for benchmarking and the model run time is 6:00 15-05-2014. This is because, for forecast dataset, it is always the latest forecast data that are intensively concerned.

| Variables | Temperature 2m above ground; Maximum temperature 2m above ground; Minimum temperature 2m above ground; Relative humidity 2m above ground; Total precipitation; Total Cloud Cover; U-Component of Wind 10m above ground; V-Component of Wind 10m above ground |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dimension count | 5 |
| Dimension Span | Longitude, latitude, forecast, ensemble, model run; (360,181,40,20,1) |
| Temporal resolution | 6 hours |
| Spatial resolution and coverage | 1 degree (111 km); Global |
| Single file size | 1.55 GB |
| Data format | NetCDF 64-bit offset |

![Figure 1. Visualization of 5D GEFS data.](image)

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

- (Q1) All ensembles for Delft at one time step: Select total precipitation for all ensembles at (4, 52) for one forecast step.
- (Q2) Forecast time series for Delft in 10 days: Select total precipitation for all ensembles at (4, 52) for 40 forecast steps.
- (Q3) 80th percentile for Delft in 10 days: Select the 80th percentile at (4, 52) from all ensembles for 40 forecast steps.
- (Q4) Ensemble mean for the Netherlands in 10 days: Select mean of 20 ensembles of total precipitation for 40 forecast steps in the bounding box (3, 51) (7, 54).

3. Benchmark environment

The benchmark test environment is constructed on the HydroNET-4 system [6] which is running on Microsoft Windows. However SciDB (version 14.3) is only available on Linux. Thus a SciDB connector is developed such that the query performance of both NetCDF and SciDB can be assessed in the same HydroNET frontend system (figure 2).

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 storage system through connectors. After data extraction, the result is returned as a binary HydroNET in-memory object to HydroNET-4. NetCDF connector to files in 64-bit offset format is based on HydroNetCDF library developed and optimized by the company. NetCDF-4 format connector is constructed on the standard library NetCDF-4.1.3. SciDB connector is constructed on shim, a super-basic SciDB client that exposes limited SciDB functionality through a simple HTTP API. All connectors can communicate with the ‘Processor’ component. 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 total 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. The server 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.
4. Results

Data used for benchmarking are stored in NetCDF 64-bit offset format, NetCDF-4 format and SciDB separately (table 2). In table 2, the last dimension in the second column is the outer most dimension to organize the storage of data. S1, S2, S3 and S4 represent 4 data schemes, i.e. structures for GEFS sample storage caused by modifying the order of dimensions. The chunk sizes of these 4 arrays keep at the same modest level, i.e. 1 value in the model run and forecast dimension, 20 values in the ensemble dimension, 181 values in the Y dimension and 360 values in the X dimension. To fully investigate whether data organization inside the chunk has an influence on query performance, GEFS sample is additionally stored into one chunk but with 4 different dimension orders, i.e. C1, C2, C3 and C4. The compression refers to DEFLATE method. All arrays store the same 8 attributes (table 1).

Table 2. Storage information of NetCDF solutions and SciDB.

| Array name       | Dimension part in the schema                                                                 | Chunk size                                                                 | Chunk count | Average chunk storage size | Total storage size |
|------------------|-----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|-------------|----------------------------|-------------------|
| NetCDF 64bit offset |                                                                                               |                                                                           |             |                            | 1.55 GB            |
| NetCDF4_S3        | (X=360, Y=181, Forecast=1, Ensembe=20, Modelrun=1)                                            | 360 x 181 x 1 x 20 x 1                                                  | 40          | 40 MB                      | 1.55 GB            |
| NetCDF4_S3_C      | (Compression)                                                                                  |                                                                           |             |                            |                   |
| SciDB_S1          | [M_idx=0:675,1:0,E_idx=0:19,20:0,F_idx=0:39,1:0,Y_idx=0:180,1:0,X_idx=0:359,360:0]          | 1 x 20 x 1 x 1 x 181 x 360                                             | 40          | 6.7 MB                     | 268 MB            |
| SciDB_S1_C        | (Compression)                                                                                  |                                                                           |             |                            |                   |
| SciDB_S2          | [M_idx=0:675,1:0,E_idx=0:19,20:0,F_idx=0:39,1:0,Y_idx=0:180,1:0,X_idx=0:359,360:0]          | 1 x 20 x 1 x 1 x 181 x 360                                             | 40          | 2.16 MB                    | 86.4 MB           |
| SciDB_S2_C        | (Compression)                                                                                  |                                                                           |             |                            |                   |
| SciDB_S3          | [X_idx=0:359,360:0,Y_idx=0:180,181:0,F_idx=0:39,1:0,E_idx=0:19,20:0,M_idx=0:675,1:0]        | 360 x 181 x 1 x 20 x 1                                                  | 40          | 6.2 MB                     | 247 MB            |
| SciDB_S4          | [E_idx=0:19,20:0,X_idx=0:359,360:0,Y_idx=0:180,181:0,F_idx=0:39,40:0,M_idx=0:675,1:0]        | 20 x 360 x 1 x 181 x 1                                                  | 40          | 5.8 MB                     | 232.4 MB          |
| SciDB_C1          | [M_idx=0:675,1:0,E_idx=0:19,20:0,F_idx=0:39,40:0,Y_idx=0:180,181:0,X_idx=0:359,360:0]       | 1 x 20 x 40 x 1 x 181 x 360                                            | 1           | 268 MB                     | 268 MB            |
| SciDB_C2          | [M_idx=0:675,1:0,E_idx=0:19,20:0,F_idx=0:39,40:0,Y_idx=0:180,181:0,X_idx=0:359,360:0]       | 1 x 40 x 181 x 360                                                     | 1           | 246.7 MB                   | 246.7 MB          |
| SciDB_C3          | [X_idx=0:359,360:0,Y_idx=0:180,181:0,F_idx=0:39,40:0,E_idx=0:19,20:0,M_idx=0:675,1:0]        | 360 x 360 x 181 x 20 x 1                                               | 1           | 246.5 MB                   | 246.5 MB          |
| SciDB_C4          | [E_idx=0:19,20:0,X_idx=0:359,360:0,Y_idx=0:180,181:0,F_idx=0:39,40:0,M_idx=0:675,1:0]        | 20 x 360 x 181 x 40 x 1                                                | 1           | 250 MB                     | 250 MB            |
Figure 4. Performance of Q2, i.e. extracting forecast time series of total precipitation.

Figure 5. Performance of Q4, i.e. calculating ensemble mean of total precipitation at Netherlands and Europe scale.

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 figure 3, 4 and 5.

It is commonly believed that retrieving values which are stored close to each other on the disk should be faster than values stored discontinuously. So considering dimension order, SciDB_S1 should take less time to execute Q1 than SciDB_S2 while SciDB_S4 should response faster than SciDB_S3. Besides, SciDB_S1 and SciDB_S4 should both response more quickly than either SciDB_S2 or SciDB_S3. This is verified in figure 3. Referring to query execution records, SciDB_S1 outperforms SciDB_S2 by 8 ms, while SciDB_S3 is slower than SciDB_S4 by 21 ms. However, the gap is not significant due to the short query execution time. To exclude the effect of chunk size on performance, GEFS dataset is managed into 4 other arrays. Then, effect of dimensions order on query performance becomes more significant (figure 3b). However, an exception appears that SciDB_C2 in which ensemble values in a single location at one forecast step are stored discontinuously responses faster than SciDB_C4 where ensemble values are stored successively. High dimensionality of GEFS dataset adds much complexity to the mapping from dimensions order to data organization on disk and this patterns is still not fully understandable.
Nevertheless, from complementary queries tested, dimension order can indeed influence query performance to a certain extent.

In figure 4, performance of NetCDF-4 solutions with schema S3 are not visualized for their high values. On the whole, 64-bit offset solution performs the best and is 9 time faster than NetCDF-4 with smaller chunk size which comes the second. SciDB arrays without compression take more time to execute the query than 64-bit offset solution, but the noise included in the measurements cannot be neglected (refer to section 3). SciDB_S1 takes less time than SciDB_S2 to process the query while the gap is insignificant. For their compressed versions, i.e. SciDB_S1_C and SciDB_S2_C, the difference becomes more obvious. The compression of SciDB arrays doubles the query response time compared with non-compressed arrays. This is also the case for NetCDF4_S3_C. The compression on NetCDF4 S5 solution however does not cause so much degradation. The high response time of NetCDF-4 S3 stores is due to little caching. Specifically, the buffered data of NetCDF4 S3 stores are flushed by caching of other NetCDF stores. While NetCDF-4 with smaller chunk size, i.e. scheme S5 can benefit from caching as presented in the figure, a possible explanation might be the chunk size of NetCDF-4 can influence the behaviour of caching as well. But more tests need to be implemented to verify it. Result of Q3 is not shown as it presents analogous patterns as Q2, but as more computation is involved, performance of SciDB experiences much degradation.

Regarding Q4, to investigate the scalability of different solutions, two locations are used for precipitation ensemble mean calculation at all 40 forecast steps (figure 5). One is the Netherlands containing 20 cells (5 x 4) and the other Europe scope has 3888 cells (72 x 54) in GEFS dataset. For the calculation of either query area, the pattern of performance keep similarity to Q2 and Q3. The effect of dimensions order is insignificant for SciDB arrays. Thanks to caching, compression of NetCDF4 S5 store cause less negative impact on query performance compared with SciDB arrays. 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, NetCDF4 S5 data stores present the best scalability and with the query area increases 200 percent, the query response time only grows 20 to 30 percent. The query processing time of 64-bit offset solution experiences a 54 percent increase. With little caching, NetCDF-4 S3 data stores cost twice the time to on the Europe scope compared to the Netherlands scope. The scalability of SciDB solutions rank the last. This is mostly because the realization of ensemble mean calculation in SciDB connector is a hybrid function using two operators, i.e. “aggregate” and “between”, which adds more complexity to query execution.

5. Conclusions and future work

Query results show that exchanging the order of dimensions inside each chunk plays significant role in query performance. But when chunk size reduces to a modest level, the impact of dimension order decreases as well. It is thus suggested that the research emphasis on chunked storage structure should move to designing appropriate chunk sizes or smartly indexing chunks instead of data structure and indexing inside chunks. Compression in most cases causes degradation especially for NetCDF-4 solutions with large chunk sizes. Caching can definitely improve performance, but the positive effect may be insignificant as cached data can be flushed by other query processes. Realization of hybrid function is a drawback for SciDB solutions, and a possible solution is to move complex calculation to the Processor, i.e. computation on top of SciDB, a hybrid solution. On the whole, NetCDF 64-bit offset solution presents the highest efficiency.

In regard to existing flaws as well as more interests inspired by the research, extension work can be conducted. Typical directions are:

- Chunk model. Previous researches [3] and this study demonstrates that the chunk size is perhaps the most crucial factor for efficiency of chunked storage structure. On the other hand, the effect of chunk size is unpredictable and for example, at which chunk size a specific query performance experience significant tuning point. Regarding this, intensive experiments can draw specific conclusions while more generic understanding relies on theoretical support. A possible way is to propose a model concerning important factors such as hardware configuration, data size and query habits to predict the performance of chunks. After this, the model should be verified by comprehensive benchmark tests.
• “Hot” and “cold” query tests. Normal approaches used to measure the average query response time include two types. “Hot” test means that the same query is executed on each data store consecutively several times and then average time is calculated. Caching effect is elaborated. As the opposite, “cold” test purges the cache every time before executing each query. Lower boundary of query response time comes from “hot” test while upper boundary depends on “cold” test. The method used in this research is a state close to “hot”, i.e. cache is not cleaned for a specific data store but can be flushed by caching of other data stores. Honestly, none of these 3 approaches reflect the reality. A more scientific way might be through analysing query logs recorded to learn what users query in reality and try to simulate those scenarios for testing.

• Parallelization. Parallel query processing of SciDB is not explored in the research. Also the research demonstrates the favorable capability of NetCDF solutions without parallelization. Then a potential topic in the future is to compare parallel query performance between SciDB and NetCDF solutions. Besides, as data loading is indeed a problem for SciDB, another direction can be parallel loading techniques.

References
[1] Rew R, Hartnett E and Caron J 2006 NetCDF-4: Software implementing an enhanced data model for the geosciences 22nd Int. Conf. on Interactive Information Processing Systems for Meteorology Oceanograph and Hydrology (Atlanta GA: USA)
[2] Brown P G 2010 Overview of SciDB: large scale array storage, processing and analysis Proc. of the 2010 ACM SIGMOD Int. Conf. on Management of Data (Indianapolis IN: USA/ACM SIGMOD Record, ACM Press) p 963-68
[3] Lee C, Yang M and Aydt R 2008 NetCDF-4 Performance Report Retrieved from http://www.hdfgroup.org/pubs/papers/2008-06_netcdfi4_perf_report.pdf
[4] Colliat G 1996 OLAP, relational, and multidimensional database systems ACM SIGMOD Record 25 p 64-9
[5] Seering A, Cudre-Mauroux P, Madden S and Stonebraker M 2012 Efficient versioning for scientific array databases Data Engineering (ICDE) 2012 IEEE 28th Int. Conf. (Arlington VA: USA) p 1013-24
[6] Reichard L, Lobbrecht A, Clark S, Catalano C, Tate B and Cox D 2014 Supporting water managers making effective decisions by using HydroNET 35th Hydrology and Water Resources Symp. (Australia) p 710-17

Acknowledgments
Financial supports for this research from the National Soft Science Research Program (2012GXS2B008), the Fundamental Research Funds for Central Public Welfare Research Institutes (CKSF2015018/KJ, CKSF2016263/KJ) and the Grant Projects of China Clean Development Mechanism Fund (2013015) are gratefully acknowledged.