

ChronosServer: fast in-situ processing of large multi-dimensional arrays with command line tools

Ramon Antonio Rodrigues Zalipynis

National Research University Higher School of Economics, Moscow, Russia
rodrigues@wikience.org

Abstract. Explosive growth of raster data volumes in numerical simulations, remote sensing and other fields stimulate the development of new efficient data processing techniques. For example, in-situ approach queries data in diverse file formats avoiding time-consuming import phase. However, after data are read from file, their further processing always takes place with code developed almost from scratch. Standalone command line tools are one of the most popular ways for in-situ processing of raster files. Decades of development and feedback resulted in numerous feature-rich, elaborate, free and quality-assured tools optimized mostly for a single machine. The paper reports current development state and first results on performance evaluation of ChronosServer – distributed system partially delegating in-situ raster data processing to external tools. The new delegation approach is anticipated to readily provide rich collection of raster operations at scale. ChronosServer already outperforms state-of-the-art array DBMS on single machine up to 193x.

Keywords: big raster data, distributed processing, command line tools, delegation approach.

1 Introduction

Raster is the primary data type in a broad range of subject domains including Earth science, astronomy, geology, remote sensing and other fields experiencing tremendous growth of data volumes. For example, DigitalGlobe – the largest commercial satellite imagery provider, collects 70 terabytes of imagery on an average day with their constellation of six large satellites [1].

Traditionally raster data are stored in files, not in databases. The European Centre for Medium-Range Weather Forecasts (ECMWF) has alone accumulated 137.5 million files sized 52.7 petabytes in total [2]. This file-centric model resulted in a broad set of raster file formats highly optimized for a particular purpose and subject domain. For example, GeoTIFF represents an effort by over 160 different remote sensing, GIS (Geographic Information System), cartographic, and surveying related companies and organizations to establish interchange format for georeferenced raster imagery [3].

The corresponding software has long been developed to process raster data in those file formats. Many tools are free, popular and have large user communities that are very accustomed to them. For example, ImageMagick is under development since

1987 [4], NetCDF common operators (NCO), a set of tools for multidimensional arrays, since about 1995 [5]; Orfeo ToolBox – remote sensing imagery processor now represents over 464,000 lines of code made by 43 contributors [6]. Many tools take advantage of multicore CPUs (e.g., OpenMP), but mostly work on a single machine.

In-situ distributed raster data processing has recently gained increased attention due to explosive growth of raster data volumes in diverse file formats. However, already existing stable and multifunctional tools are largely ignored in this research trend. Thus, raster operations are re-implemented almost from scratch delaying emergence of a mature in-situ distributed raster DBMS.

This paper describes the prototype extension of ChronosServer [7, 8] leveraging existing command line tools for in-situ raster data processing on a computer cluster of commodity hardware. Unlike current systems, it is easier and faster to equip ChronosServer with wide variety of raster operations due to new delegation approach. Thus, it is anticipated that it is possible to quickly develop new distributed file-based raster DBMS with rich functionality and exceptional performance.

2 In-situ raster data processing

In-database data storage (in-db, import-then-query) requires data to be converted (imported) to internal database format before any queries on the data are possible. Out-of-database (out-db, in-situ, file-based, native) approach operates on data in their original (native) file formats residing in a standard filesystem without any prior format conversions.

2.1 State-of-the-art

PostgreSQL extensions PostGIS [9] and RasDaMan [10] work on single machine and allow registering out-database raster data in file system in their native formats. Enterprise RasDaMan version claims to be in-situ enabled and distributed, but is not freely available [11]. PostGIS has poor performance on multidimensional arrays (e.g. NetCDF, HDF or Grib formats [12]). No performance evaluation has been ever published for enterprise RasDaMan. SAGA [13] executes only distributed aggregation queries over data in HDF format. SWAMP [14] accepts shell scripts with NCO and parallelizes their execution. Hadoop extensions SciHadoop [15] and SciMATE [16] were never released publicly. They implement drivers reading Hadoop DFS chunks as if they are in HDF or NetCDF formats. Galileo [17] indexes geospatial data with distributed geo-hash. SWAMP launches command line tools but focuses on NCO and requires scripts looping over files with explicitly specified file names. The proposed approach is universally applicable to any tool and abstracts from “file” notion at all.

Commercial ArcGIS ImageServer [18] claims in-situ raster processing with custom implementation of raster operations. However, in a clustered deployment scenario all cluster nodes are recommended to hold copies of the same data or fetch data from a centralized storage upon request what negatively impacts scalability. Commercial Oracle Spatial [19] does not provide in-situ raster processing [20]. Open source

SciDB is specially designed for distributed processing of multidimensional arrays [21]. However, it does not operate in-situ and imports raster data only converted to CSV format – very time-consuming and complex undertaking. Moreover, SciDB lacks even core raster operations like interpolation which makes it an immature and not widely used product [22]. Intel released open source TileDB on 04 Apr. 2016. It is yet neither distributed nor in-situ enabled [23].

Hadoop [24] and experimental SciDB streaming [25] allow launching a command line tool, feed text or binary data into its standard input and ingesting its standard output. Note two time-consuming data conversion phases in this case: data import into internal database format and their conversion to other representation to be able to feed to external software. The proposed approach directly submits files to external executables without additional data conversion steps.

SciQL was an effort to extend MonetDB with functionality for processing multidimensional arrays [26]. However, it has not yet finished nor its active development is seen so far. Also, SciQL does not provide in-situ raster processing.

2.2 In-situ approach benefits

This section collects in one place advantages and challenges of in-situ approach that are quite scattered in the published literature.

- *Avoid inefficient neighborhood.* Traditionally, BLOB (Binary Large Object) data type served for in-db raster storage (PostGIS, RasDaMan). Physical layouts where raster data are close to other data types are quite inefficient since the former are generally much larger than the latter.
- *Leverage powerful storage capabilities.* Some raster file formats support chunking, compression, multidimensional arrays, bands, diverse data types, hierarchical namespaces and metadata. These techniques are fundamental for raster storage; their implementation for an emerging in-db storage engine results in yet another raster file format.
- *Avoid conversion bottleneck.* In mission-critical applications it is important to be able to analyze the data before their new portion arrives or a certain event happens. In some cases the conversion time may take longer than the analysis itself. The data arrival rate and their large volumes may introduce prohibitively high conversion overheads and, thus, operational failure.
- *Avoid additional space usage.* Most data owners never delete source files after any kind of format conversions including database import. There are numerous reasons for this including unanticipated tasks that may arise in future that are more convenient, faster, easier or possible to perform on the original files rather than their converted counterparts. Storing both source data and their in-db copies requires additional space that may be saved by in-situ approach.
- *Reduce DBMS dependence.* It is easier to migrate to other DBMS keeping data in a widely adopted storage format independent from a DBMS vendor.

- *Leverage other software tools (this paper).* Out-db raster data in their native formats remain accessible by any other software which was inherently designed to process file-based data.

Key difficulties lie in the ability to perform the same set of operations on data in different file formats. Three data models are most widely used that allow abstracting from file format: Unidata CDM, GDAL Data Model, ISO 19123 (not cited due to space constraints). Most existing command line tools use those models and, thus, are capable to handle data in diverse raster file formats.

3 ChronosServer architecture

3.1 Raster data model: abstracting from files, their locations and formats

This paper focuses on climate reanalysis and Earth remote sensing global gridded raster data represented as multidimensional arrays and usually stored in NetCDF, Grib and HDF file formats. For example, AMIP/DOE Reanalysis 2 (R2) spans several decades, from 01.01.1979 to current date with 6 hour time interval and contains over 80 variables [27]. The grid resolution is usually $2.5^\circ \times 2.5^\circ$. Global grids for each variable are stored in a sequence of separate files partitioned by time. File names contain variable codename, e.g. files with surface pressure are named `pres.sfc.1979.nc`, `pres.sfc.1980.nc`, ..., `pres.sfc.2015.nc`. Where “`pres.sfc`” denotes surface pressure, 1979 is year, “.nc” is NetCDF file extension. Usually file naming is much more complex (e.g., compare to AIRS/AMSU daily file name for CO₂ satellite data: `AIRS.2004.08.01.L3.CO2Std001.v5.4.12.70.X09264193058.hdf`). Note, that the data are usually already split by files by data providers.

ChronosServer distributes files among cluster nodes without changing their names and formats. Any file is always located as a whole on a machine in contrast to parallel or distributed file systems. It introduces a data model to work with grids, not files to abstract from “file” notion, file naming, their locations, formats and other details that are unique to every dataset and not relevant for data analysis. ChronosServer dataset namespace is hierarchical. For example, “`r2.pressure.surface`” refers to surface pressure of R2 reanalysis. ChronosServer provides SQL-like syntax for sub-setting grids. For example, “`SELECT DATA FROM r2.pressure.surface WHERE TIME_INTERVAL = 01.01.2004 00:00 - 01.01.2006 00:00 AND REGION = (45, 60, 50, 70)`” returns time series of R2 surface pressure in the specified time interval and region between 45°S – 50°N and 60°W – 70°E. The query execution may involve several cluster nodes. Any grid or time series from any dataset may be extracted with the same syntax regardless of original file format, file split policy and other details.

3.2 Cluster orchestration

ChronosServer cluster consists of workers launched at each node and a single gate at a dedicated machine. Gate receives client queries and coordinates workers responsible for data storage and processing. All workers have the same hierarchy of data directo-

ries on their local filesystems. A worker stores only a subset of all dataset files and only a portion of the whole namespace relevant to the data it possesses. A file may be replicated on several workers for fault tolerance and load balancing. It is not required to keep all workers up and running for the whole system to be operational.

The gate is unaware of file locations until a worker reports them to it. This is done for better scalability and fault tolerance. Upon startup workers connect to gate and receive the list of all available datasets and their file naming rules. Workers scan their local filesystems to discover datasets and their time intervals by parsing dataset file names. Workers transmit to gate the list of time intervals for each dataset they store. Gate keeps this information in worker pool – in-memory data structure used during query planning that maps time intervals to their respective owners (workers).

4 New delegation approach

4.1 ChronosServer raster data processing commands and their distributed execution

ChronosServer syntax of a raster data processing command is the same as launching a tool from a command line. Command names coincide with names of existing command line tools. ChronosServer command options have the same meaning and names as for the tool but without options related to file names or paths. Commands and tools also support options with long names having the same meaning.

```
ncatted [-a ...] [--bfr sz] [-D nco_dbg_lvl] [--glb ...] [-h]
        [--hdr_pad nbr] [-l path] [-O] [-o out.nc] [-p path] [-R]
        [-r] [-t] in.nc [[out.nc]]
-a,    variable_name,mode,attribute_type,attribute_value
        mode = a,c,d,m,o (append, create, delete, modify, overwrite)
        att_typ = f,d,l/i,s,c,b (float, double, long, short, char, byte)
--bfr_sz, --buffer_size sz          Buffer size to open files with
-D, --dbg_lvl, --debug-level lvl      Debug-level is lvl
--glb nm=val                        Global attribute to add
-h, --hst,                          Do not append to "history" global attribute
--hdr_pad                          Pad output header with nbr bytes
-l, --lcl                          Local storage path for remotely-retrieved files
-o, out.nc                          Output file name (or use last argument)
-O, --ovr                          Overwrite existing output file, if any
-p, --path path                    Path prefix for all input filenames
-R, --rtn                          Retain remotely-retrieved files after use
-r, --revision                      Compile-time configuration and program version
-t, --typ_mch,                      Type-match attribute edits
in.nc [[out.nc]] Input file name [[Output file name]]
```

Fig. 1. Parameters of the NCO ncatted tool (all listed) and Chronos ncatted command (in bold)

For example, NCO consists of several standalone command line tools: `ncap2` (Arithmetic Processor v.2), `ncks` (Kitchen Sink), `ncatted` (Attribute Editor – metadata manager, Fig. 1), etc. [28, 29]. Metadata are crucial component of any raster data. NetCDF and many other formats store metadata as attributes (key-value pairs). For example, attribute named “_FillValue” holds a constant used to mark raster cells with missing values (e.g., -9999).

By default, command is applied to the whole available dataset time interval and spatial coverage. They may be restricted by “select” query with alias dataset name specification. New virtual dataset will contain subset of the original dataset. Its name (alias) may be used in the subsequent commands. It is helpful to test a series of commands on a dataset sample to check hypotheses about the anticipated results before submitting large-scale query involving large data volumes to save time.

For example, ChronosServer command for “_FillValue” attribute deletion from dataset “r2.pressure.surface” is “`ncatted -a _FillValue,r2.pressure.surface,d,,`”. Instead of “variable_name” – the term specific for NetCDF format, ChronosServer `ncatted` accepts a dataset name to be independent of a concrete format. Usually the same attributes are duplicated in all files of a dataset.

The gate receives and parses command line options, verifies their correctness and absence of malicious instructions since they are passed to operating system shell. The dataset or its subset is locked for reading/writing depending on the command. Several commands may work concurrently if they do not block each other. Gate selects workers on which dataset files with the required time/space intervals are located and sends them the modified command (see below). Workers complement command line with full paths to dataset files according to time and space limitations and launch the tool on each file.

In the simple case above, ChronosServer invokes several instances of the NCO `ncatted` tool on the cluster nodes where at least one dataset file is located (`pres.sfc.1979.nc`, ..., `pres.sfc.2015.nc`). The execution command line for file `pres.sfc.1979.nc` is “`<path to ncatted.exe> -a missing_value,pres,d,, <data path>\pres.sfc.1979.nc`”. The file path and “pres” were automatically put by worker and gate correspondingly. The latter is the NetCDF variable name that stores R2 surface pressure (NetCDF3 format does not have hierarchical namespace and stores data in structures called “variables”).

Workers also collect standard output of the tool which is sent to gate after its completion. Running tool on a different cluster node in case of a hardware failure is under development. Gate reports to the user once it receives success messages from all workers involved in the command execution. Report contains the merged standard outputs from each run of the tool and total elapsed time.

4.2 Distributed apply-combine-finally execution scheme (under development)

Raster operations can be broadly classified as global (involve all data), local (pixel-wise), focal (cell values from a rectangular window are required to compute new cell value), zonal (same as focal but spatial region is defined by a function) [30]. Thus, some operations cannot be completed autonomously using data on a single cluster

node. For example, R2 data interpolation for 1980 year from 6 to 3 hour time step requires grids for December 1979 and January 1981. Also, computation of maximum mean winter pressure involves all files potentially located on different cluster nodes.

In this case, user specifies commands: APPLY command1 INTERVALS intervals COMBINE command2 FINALLY command3. ChronosServer ensures that files on each involved node contain data in given temporal and spatial intervals (e.g., in case of winter means, there should be nodes with data for all winter months for at least two consecutive years: 1980-1981, 1981-1982, 1982-1983, etc. to be able to compute the mean). This may require data movement between cluster nodes. After the intervals requirement is met, command1 is executed autonomously on the data intervals on corresponding cluster nodes. Since some nodes may have several disjoint intervals, their intermediate results may be combined on the same node to reduce network traffic with command2 if it is possible (e.g. compute maximum of the means of intervals 1980-1981 and 1982-1983 that happened to reside on the same node). All results are gathered on a single node and command3 is applied to obtain final result (e.g., find maximum of means or maximums of maximums if combine phase was applied).

Unlike existing schemes [31, 32], the proposed distributed execution scheme takes into account peculiarities inherent to raster operations, geospatial data and ChronosServer file-based storage model. For example, respective intervals must be specified to guarantee the raster operation (command1) is possible to accomplish within a single node. It is widely recognized that numerous data processing tasks are much easier to parallelize once actions are expressed in functional style, not “for” loops.

4.3 Benefits of the proposed delegation approach

While in-situ approach leverages benefits of already existing sophisticated file formats, delegation approach leverages benefits of already existing standalone command line tools.

- *Avoid learning new language.* ChronosServer provides command line syntax that is well-known to every console user instead of a new SQL dialect.
- *Steep learning curve.* Users work with ChronosServer as if with console tools they have accustomed to with only minor changes to already familiar tools’ options.
- *Documentation reuse.* Most of the tool’s documentation is applicable to the corresponding ChronosServer command due to exactly the same meaning and behavior.
- *Output conformance.* Output files are formatted as if a tool was launched manually.
- *Language independence.* ChronosServer may use tools written in any programming language.
- *Community support.* Bugs in tools are fixed by their developers as well as new functionality added, usage suggestions via mail lists are obtained regardless of ChronosServer context.
- *Zero-knowledge development (0-know dev.).* Developers of existing and emerging tools do not have to know anything about ChronosServer in order the tool could be used in ChronosServer.

The main difficulty of the proposed approach lies in the correct specification of the “intervals”, “combine” and “finally” clauses. Meta-commands are a possible simplification of the problem. They consist of a single command line which translates to pre-defined apply-combine-finally clauses.

5 Performance evaluation

To date, the only freely available distributed raster DBMS is SciDB, not operating in-situ. It lacks many core raster operations. Thus, only the performance of some basic ChronosServer and SciDB raster operations are compared. It is of special interest to evaluate ChronosServer against SciDB since the latter is currently being most actively popularized among similar DBMS at top journals and conferences [33–35].

Test of source data volume comprised only 100.55 MB in NetCDF3 format since it is impossible to import large data volumes into SciDB in a reasonable time frame (section 5.2). In addition, experiments were carried out on a single machine for two reasons. First, raster operations being evaluated have linear scalability. Increasing machine number by a factor of N should roughly increase the performance also by N . Second, unlike ChronosServer, SciDB cluster deployment is very labor-intensive. Comparison of both systems running on computer cluster is left for future work.

However, small test data volume and single machine turned out to be sufficient for representative results (Table 1). Table 1 summarizes experimental results while details are given in following subsections.

Table 1. ChronosServer and SciDB performance comparison

Operation	Execution time, seconds			Ratio, SciDB / ChronosServer	
	SciDB	ChronosServer			
		Cold	Hot	Cold	Hot
Data import	720.13	19.82	7.96	36.33	90.47
Max	13.46	4.43	3.10	3.04	4.34
Min	12.87	4.71	3.33	2.73	3.86
Average	21.42	4.71	3.23	4.55	6.63
Wind speed calc.	25.75	3.50	2.10	7.36	12.26
Chunk 100×20×16	56.19	1.68	0.374	33.45	150.24
Chunk 10×10×8	222.11	1.98	1.15	112.18	193.14

5.1 Test raster data and experimental setup

Eastward (U-wind) and northward (V-wind) wind speed (Fig. 2) at 10 meters above surface from NCEP/DOE AMIP-II Reanalysis (R2) were used for experiments [27]. These are 6-hourly forecast data (4-times daily values at 00.00, 06.00, 12.00 and 18.00). Data are 3-dimensional on 94 latitudes \times 192 longitudes Gaussian grid in NetCDF3 format. Dataset does not contain missing values. Single file is approximately 50.2 GB; total data volume is 3.63 GB for the whole available time interval 1979 – 2015. Due to SciDB limitations (section 5.2), only data for 1979 year were used (about 100.55 MB as mentioned earlier).

Both ChronosServer and SciDB were run on Ubuntu 14.04 inside VirtualBox on Windows 10. Note that ChronosServer is also capable to run natively on Windows, unlike SciDB. The machine is equipped with SSD (OCZ Vertex 4). VirtualBox was assigned 4 GB RAM and 2 CPU cores (Intel Core i5-3210M, 2.50 GHz per core). SSD speed inside VirtualBox: 4573.28 MB/sec and 222.04 MB/sec (cached and buffered disk reads respectively as reported by hdparm utility); 350 MB/sec disk write as reported by dd utility.

ChronosServer has 100% Java code, ran one gate and one worker, Java 1.7.0_75, OpenJDK IcedTea 2.6.4 64 bit, max heap size 978 MB (-Xmx), NCO v4.6.0 (May 2016). SciDB is mostly written on C++, v15.12 was used (latest, Apr. 2015) with recommended parameters: 0 redundancy, 4 instances per machine, 4 execution and prefetch threads, 1 prefetch queue size, 1 operator threads, 128 MB array cache, etc.).

Two types of query runs were evaluated: cold (query executed first time on given data) and hot (repeated query execution on the same data). Time reported in Table 1 is the average of three runtimes of the same query. Respective OS commands were issued to free pagecache, dentries and inodes each time before executing cold query to prevent data caching at various OS levels. Table 1 does not report cold and hot runs for SciDB since it did not reveal any significant difference in runtime between them. In contrast, ChronosServer does not cache data but benefits from native OS caching and demonstrates significant speedup for hot runs. This is particularly useful for continues experiments with the same data. The need for this type of experiments occurs quite often (e.g., tuning certain parameters, refer to section 5.5 for an example).

5.2 SciDB data import and ChronosServer data discovery

Importing data into SciDB involves considerable efforts on software development for each dataset being considered. SciDB does not yet provide out-of-the-box import tool from formats other than CSV. The overall import procedure is very time-consuming and error-prone (both due to complicated raster formats and related possible coding bugs as well as inherent floating point calculations). Lack of documentation and complex query syntax (Appendix B) may elongate data import for several weeks.

For SciDB “data import” row, Table 1 reports only time taken to automatically import U-wind speed data for 1979 year (50.2 MB) from NetCDF3 format into SciDB. It does not report the time spent for Java program development to actually perform the import. Importing V-wind speed for 1979 also takes approximately the same amount

of time. Estimate time is 14.76 hours to automatically import U- and V-wind speed for 1979 – 2015. Thus, only U- and V-wind speed vectors for 1979 are considered for performance evaluation in the next subsections. This small data sample turned out to be representative for convincing results.

SciDB data import from NetCDF3 included reading original data file, preparing string representation of 94×192 grid for each time step in a format ingestible by SciDB, saving string to CSV file, and invoking SciDB tool to import grid from CSV.

On the contrary, to add new data under ChronosServer management, it is sufficient to copy data files on a cluster node and add a short entry in ChronosServer XML file specifying rules for file naming and a handful of some other information. ChronosServer will discover files as described in section 3.1. Worker discovers files at startup. Table 1 “Data import” row for ChronosServer reports time of its “cold” and “hot” startup. The former startup mode rediscovers completely from scratch all existing as well as any newly added data. The latter mode assumes no new data were added since previous startup. Both measured times include complete startup time of one gate and one worker, metadata transfer from gate to worker (registered datasets), data discovery by worker, logging and any other startup overhead.

ChronosServer is able to discover 803 datasets with total volume of 6.78 GB in file formats NetCDF-3, -4, HDF-4, -5, Grib-2 (a collection of diverse satellite and climate reanalysis products) in 20 and 8 seconds for cold and hot startups respectively. This is 36x and 90x faster than SciDB imports just 50.2 MB of data.

5.3 Simple statistics

Table 1 rows for max, min and average report time taken by the systems to calculate maximum, minimum and average U-wind speed for 1979 year for each 94×192 grid cell. Computation involves traversing 1460 time steps. ChronosServer is about 3 to 6 times faster than SciDB.

5.4 User-defined arithmetic expressions

Both ChronosServer and SciDB support user-defined arithmetic expressions that could be applied to raster data. As an example, wind speed (ws) at each grid cell and time point is calculated from its eastward (u) and northward (v) components as $ws = \sqrt{u^2 + v^2}$ (Fig. 2).

In this case, ChronosServer is 7 to 12 times faster than SciDB (Table 1).

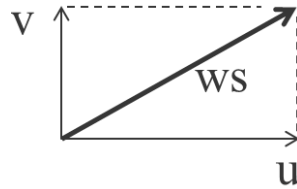


Fig. 2. Wind speed vector (ws) and its eastward (u) and northward (v) speed vectors.

It is worth noting, that SciDB query for wind speed calculation is very complex (Appendix B), unlike that for ChronosServer (Appendix A).

5.5 Multidimensional chunking

Chunking is the process of partitioning original array (raster) onto a set of smaller subarrays called chunks (Fig. 3). Chunks are autonomous, possibly compressed arrays with contiguous storage layout. A chunk is usually read/written completely from/to disk in one request to storage subsystem. Chunking is one of the classical approaches to significantly accelerate disk I/O when only a portion of raster is read. Consider reading a 6×2 slice from a 2D array (Fig. 3). For a row-major storage layout, two vertically adjacent cells are located far apart each other. A possible solution is to read 6 portions sized 1×2 which requires 6 I/O requests and disk seeks (Fig. 3a). For a compressed array, much larger part of it might be required to be read and uncompressed before getting the requested portion. In contrast, only chunks containing required data are read from disk from a chunked raster. However, inappropriate chunk shape may result in large I/O overhead (Fig. 3b). Good chunk shape allows to reduce communication with storage layer, disk seeks and I/O volume (Fig. 3c).

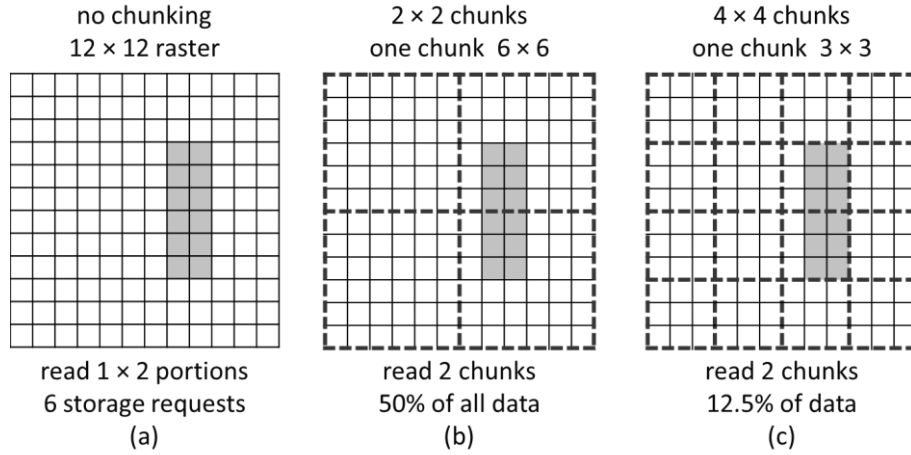


Fig. 3. Chunking: row-major storage layout, read 6×2 slice

Since many raster operations are mostly I/O bound [28], chunk shape is one of the crucial performance parameters for a dataset [34]. Chunk shape depends on data characteristics and workload. Optimal chunk shape usually does not exist for all access patterns. It is also difficult to guess good chunk shape a priori: chunk shape is often tuned experimentally. Thus, raster DBMS must be capable to quickly alter chunk shape in order to support experimentation as well as to adapt to dynamic workloads.

To estimate chunking speed of both systems, U-wind speed data for 1979 were chunked with two different chunk shapes $ts \times lats \times lons$: $100 \times 20 \times 16$ and $10 \times 10 \times 8$, ts , $lats$ and $lons$ are chunk sizes along time, latitude and longitude axes respectively.

ChronosServer is up to 193 times faster than SciDB (Table 1). Presented timings are the average for 3 consecutive runs as mentioned earlier (each cold run precedes OS cache clear). In practice, ChronosServer could be even faster: 973 ms execution time (less than a second) could be obtained for a hot run leading to 228.3x speedup.

6 Conclusion

The paper presented new approach of delegating in-situ raster data processing to existing command line tools. The approach has numerous benefits and is under development as an extension to ChronosServer – inherently distributed, file-based system for high performance raster data dissemination [7, 8]. This paper also presented first results on performance evaluation of ChronosServer against SciDB – one of the most popular, distributed state-of-the-art raster DBMS [33–35]. Raster operations were executed on 100.55 MB wind speed data from NCEP/DOE AMIP-II Reanalysis. This was governed by SciDB which is unable to import large data volumes in a reasonable time frame. However, this small data sample turned out to be sufficient for representative comparison. ChronosServer always outperforms SciDB. Also, query syntax of ChronosServer is much easier and cleaner compared to SciDB. Max, min and average operations are 3x to 6x faster, user-defined arithmetic expression was shown to be 7x to 12x faster while altering 3D chunk shape is about 33x to 228.3x faster.

Acknowledgements. This work was partially supported by Russian Foundation for Basic Research (grant #16-37-00416).

A Appendix. ChronosServer queries

Max U-wind speed (section 5.3):

```
ncap2 -alias u,r2.wind.10m.u
      -alias umax,r2.wind.10m.uv.umax
      -s "$ (umax)=$ (u) .max ($time) "
```

Calculate wind speed (section 5.4):

```
ncap2 -alias u,r2.wind.10m.u
      -alias v,r2.wind.10m.v
      -alias ws,r2.wind.10m.uv.ws
      -s "$ (ws)=sqrt ($ (u) *$ (u) + $ (v) *$ (v) ) ; "
```

Alter chunk shape to 10×10×8 (section 5.5):

```
ncks -4 --cnk_map dmn --cnk_plc g2d --cnk_dmn time,10
      --cnk_dmn lat,10 --cnk_dmn lon,8
      r2.wind.u10m.u r2.wind.u10m_ch10x10x8
```

B Appendix. SciDB queries

Initial SciDB array for U-wind speed:

```
r2_u10m<value:float>
[time=0:*,1,0,lat=0:93,94,0,lon=0:191,192,0]
```

Max U-wind speed (section 5.3):

```
store(aggregate(r2_u10m, max(value), lat, lon),
r2_u10m_max);
```

Calculate wind speed (section 5.4):

```
store(project(apply(join(r2_u10m, r2_v10m), ws,
float(sqrt(r2_u10m.value * r2_u10m.value + r2_v10m.value
* r2_v10m.value))), ws), r2_ws10m);
```

Alter chunk shape to 10×10×8 (section 5.5):

```
store(redimension(r2_u10m, <value:float>
[time=0:*,10,0,lat=0:93,10,0,lon=0:191,8,0]),
r2_u10m_10x10x8);
```

According to the answer of SciDB developers on their forum (question posted by the author of this paper in August 2016), above query is currently the fastest way to alter chunk size in SciDB: <http://forum.paradigm4.com/t/fastest-way-to-alter-chunk-size/>

References

1. Launching DigitalGlobe's Maps API | Mapbox, <https://www.mapbox.com/blog/digitalglobe-maps-api/>
2. Grawinkel, M. et al.: Analysis of the ECMWF Storage Landscape. In: 13th USENIX Conference on File and Storage Technologies, p. 2, February 16–19 (2015), Santa Clara, CA, https://usenix.org/system/files/login/articles/login_june_18_reports.pdf
3. GeoTIFF, <http://trac.osgeo.org/geotiff/>
4. ImageMagic: History. <http://imagemagick.org/script/history.php>
5. NCO Homepage, <http://nco.sourceforge.net/>
6. The Orfeo ToolBox on Open Hub, <https://www.openhub.net/p/otb>
7. Rodrigues Zalipynis, R.A.: ChronosServer: real-time access to “native” multi-terabyte retrospective data warehouse by thousands of concurrent clients. Informatics, cybernetics and computer engineering, vol. 14 (188), pp. 151–161 (2011)
8. ChronosServer, <http://www.wikience.org/chronosserver/>
9. Chapter 5. Raster Data Management, Queries, and Applications, http://postgis.net/docs/manual-2.2/using_raster_dataman.html
10. Baumann, P., Dumitru, A., Merticariu, V.: The Array Database That Is Not a Database: File Based Array Query Answering in rasdaman. SSTD 2013, LNCS, vol. 8098, pp. 478–483, Springer, Heidelberg (2013)

11. RasDaMan features, <http://www.rasdaman.org/wiki/Features>
12. NetCDF. <http://www.unidata.ucar.edu/software/netcdf/docs/>
13. Wang, Y., Nandi, A., Agrawal, G.: SAGA: Array Storage as a DB with Support for Structural Aggregations. In: SSDBM'14, June 30 - July 02, (2014)
14. Wang, L. et al.: Clustered workflow execution of retargeted data analysis scripts. In: CCGRID 2008
15. Buck, J.B., Watkins, N., LeFevre, J., Ioannidou, K., Maltzahn, C., Polyzotis, N., Brandt, S.: SciHadoop: Array-based Query Processing in Hadoop. In: Proc. of SC (2011)
16. Wang, Y., Jiang, W., Agrawal, G.: SciMATE: A Novel MapReduce-Like Framework for Multiple Scientific Data Formats. In: Proc. of CCGRID, pp. 443–450, May (2012)
17. Malensek, M., Pallickara, S.: Galileo: A Framework for Distributed Storage of High-Throughput Data Streams. In: Proc. of the 4th IEEE/ACM Intl. Conf. on Utility and Cloud Computing (2011).
18. ArcGIS for Server | Image Extension, <http://www.esri.com/software/arcgis/arcgisserver/extensions/image-extension>
19. Oracle Spatial and Graph, <http://www.oracle.com/technetwork/database/options/spatialandgraph/overview/index.html>
20. Georaster: Import very large images with sdo_ge... | Oracle Community, <https://community.oracle.com/thread/3820691?start=0&tstart=0>
21. Paradigm4: Creators of SciDB, <http://scidb.org/>
22. Interpolation - SciDB usage - SciDB Forum, <http://forum.paradigm4.com/t/interpolation/1283>
23. TileDB - Scientific data management made fast and easy, <http://istc-bigdata.org/tiledb/index.html>
24. Hadoop Streaming, wiki.apache.org/hadoop/HadoopStreaming
25. GitHub - Paradigm4/streaming: Prototype Hadoop streaming-like SciDB API, <https://github.com/Paradigm4/streaming>
26. Zhang, Y. et al.: SciQL, Bridging the Gap between Science and Relational DBMS. In: IDEAS11, September 21-23, Lisbon, Portugal (2011).
27. NCEP-DOE AMIP-II Reanalysis, <http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis2.html>
28. Zender, C.S.: Analysis of self-describing gridded geoscience data with netCDF Operators (NCO). Environmental Modelling & Software, vol. 23, pp. 1338–1342 (2008)
29. Zender, C.S., Mangalam, H.: Scaling properties of common statistical operators for gridded datasets. Intl. J. of High Performance Computing Applications, vol. 21 (4), pp. 458–498 (2007).
30. Geospatial raster data processing, http://rgeo.wikience.org/pdf/slides/rgeo-course-04-raster_processing.pdf
31. Wickham, H.: The split-apply-combine strategy for data analysis. Journal of Statistical Software, vol. 40, pp. 1–29 (2011)
32. Yang, H.C., Dasdan, A., Hsiao, R.L., Parker, D.S.: Map-reduce-merge: Simplified relational data processing on large clusters. In: ACM SIGMOD, June 12–14, Beijing (2007)
33. Stonebraker, M., Brown, P., Zhang, D., Becla, J.: SciDB: A Database Management System for Applications with Complex Analytics, Comput. Sci. Eng., vol. 15 (54), (2013)
34. Cudre-Mauroux, P., et al.: A demonstration of SciDB: A science-oriented DBMS. Proceedings of VLDB, Endowment, vol. 2 (2), pp. 1534–1537 (2009)
35. Planthaber, G., Stonebraker, M., Frew, J.: EarthDB: scalable analysis of MODIS data using SciDB. In: BigSpatial, pp. 11–19, (2012)