# An OPeNDAP server at the peta-byte scale

**Guillaume PERNOT[(1)], Jean-Christophe BABINET[(2)]**

***Auvéa Ingénierie***

*100, route de Francazal – 31120 Portet-sur-Garonne*

*[(1)] EMail: gpernot@praksys.org*

*[(2)] EMail: jc.babinet@auvea.fr*

## ABSTRACT

With respect to the rise of geo-referenced databases, their increased complexity and their upsurge in size, this contribution describes DfP, an OPeNDAP server able to serve petabytes of heterogeneous geo-referenced data in a homogeneous, quality-controlled, performance-guaranteed, and standard-conformant way.

Keywords: GIS, petabyte-scale databases, OPeNDAP, OAIS, geoscience, time-series, metadata

## INTRODUCTION

Thanks to new missions, new captors and model refinements, geosciences are producing more and more data. It is admitted that, on average, data production rate doubles each year [1]. Moreover, these data are heterogeneous, as they cover various fields like oceanography, geology, atmospheric sciences, seismology, etc...

Geosciences have reached a point where storing, accessing and sharing the data they produce causes concern.

With respect to access and sharing, during the last decade, tremendous progresses have been achieved in metadata catalogs interoperability through standardization efforts: Open GeoSpatial Consortium standards, ISO-19100 family, INSPIRE, CCSDS ([2], [3]), to name a few.

On the other hand, storage itself is still a challenge.

It is now not uncommon to have to deal with hundreds of terabytes or even petabytes of data. At this scale, data is spanned over hundreds or thousands of disks, and this brings new issues. Requests atomicity, database consistency, failure tolerance, system administration, overall system performance and global power consumption have to be carefully examined[1].

This note describes DfP, an OPeNDAP [8] server currently developed at Auvéa Ingénierie able to serve petabytes of geo-referenced heterogeneous data (*§ Service presentation*). A first proof-of-concept has been implemented [9], and some challenges and development perspectives are exposed (*§ Challenges*) .

## SERVICE PRESENTATION

Target users community is geoscientists at large, be there oceanographers, climatologists, seismologist, geologists, biogeochemists, etc... In these fields, datasets are geo-referenced, multi-dimensional and array-oriented (including images).

Geo-referenced data is any data that has a geographical extent. That said, it has a wide variety of flavors. Structures, formats, spatial reference systems, types, units, precisions, parameter names, etc... vary across application fields and thus databases.

[1] Non-technical aspects such like system security, copyrights or intellectual property of databases are out of the scope of this paper and are intentionally left apart, even though they are obviously taken into account during implementation.

As it is a quite standard process in geosciences, we will consider *databases* are incrementally built by adding *datasets*. All datasets in a database share a common structures: parameter names, format, types, etc... A model run or satellites sensors, for example, append datasets as they are computed or acquired, respectively.

OPeNDAP is a protocol to access databases of multi-dimensional arrays. While having some limitations [4], it is a *de facto* standard, as it is widely used in target communities, and is available in a large spectrum of applications: virtually all GIS applications (through GDAL bindings), MATLAB/Octave, IDL, GrADS, FERRET, OceanDataView, GoogleEarth...

Current OPeNDAP implementations focus on serving *datasets*, not *databases*. This can be awkward for users, as they need to know and understand how (and sometimes, when), datasets are produced to extract data they need, and, still, the process can be cumbersome. Also, from the producer point of view, scalability is an issue.

Proposed OPeNDAP service, beyond simply aggregating datasets, makes data easily and conveniently accessible, and is able to serve databases at the petabyte scale. This paper focuses on the "Archival Storage", "Data Management", and "Preservation Planning" functional entities in the OAIS functional Model (*c.f.* figure 1).

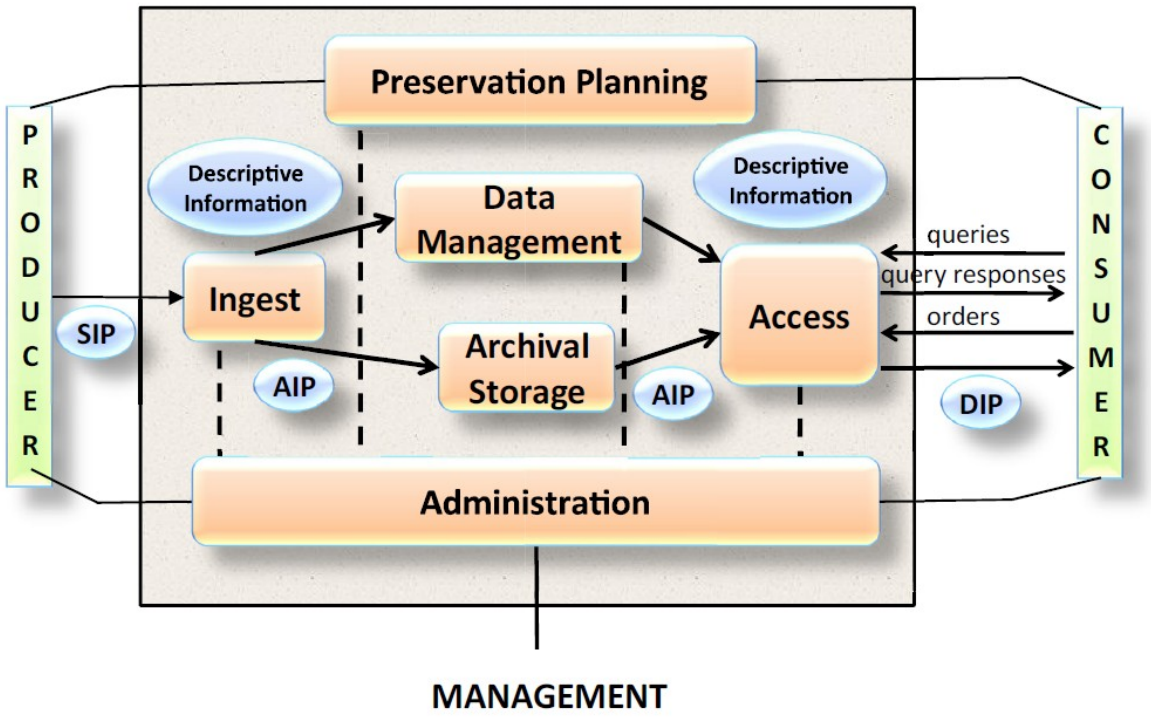


Figure 1: OAIS Functional Model (from [3])

All along conception, we adopted "pragmatic design": privileging today solutions over promises for tomorrow. That is, implementation is running with current softwares on current hardware, using current file formats and protocols.

Briefly put, DfP holds metadata and geographical extent of each dataset in a relational GIS-enabled database, classically, and store data themselves in a sharded key-value database. On request, an orchestrator queries the RDBMS that returns datasets that *may* have values corresponding to selection, then query dataset servers to introspect each dataset for finer results, and finally aggregate results (figure 2).

Preliminary tests [5] were done using a 2TB database of global ¼-degree model output from Finite-State Lyapunov Exponents as describe in [6] and [7].

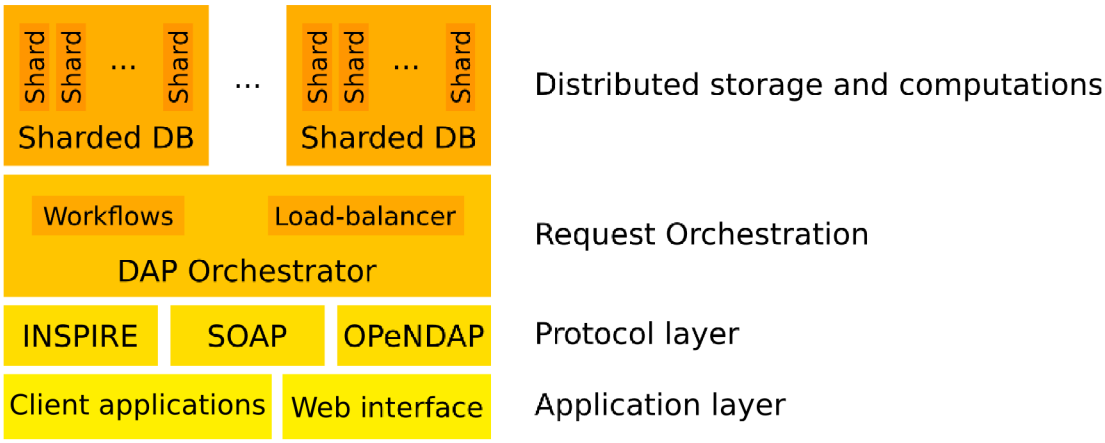

Figure 2: Service bloc diagram

# CHALLENGES

## Usability

First things first, accessing data means being able to conveniently describe wanted data, and retrieve it in an acceptable time-frame and in a practical format.

- Query syntax and web interface

Hosted databases can be queried with DAP syntax. A DAP query consist of a *projection* and a *selection* [8]. *Projection* is a list of parameters the user wants to retrieve. *Selection* describes the geographical extent, date range, and any other constraints.

For example, say we have the following databases available:

- AIR, with a parameter "temperature", that holds air temperature at the given location;
- MODIS [11], with a parameter "snow" set to 1 if snow coverage is detected, 0 otherwise;
- COUNTRY, with a parameter "name" that, for given latitude and longitude, holds the name of the country.

Say user wants localized air temperature (*i.e.* projection is "longitude,latitude,date,AIR.temperature") in bounding box with corner coordinates (0.77, 42.59) and (1.66, 43.21), from 1st January 2000 to 1st January 2001, but only where snow was detected and limit results to France. The DAP query will be:

```
longitude,latitude,date,AIR.temperature&latitude<43.21&latitude>42.59&longitude<1.66&
longitude>0.77&date>20000101&date<20010101&MODIS.snow=1&COUNTRY.name=FRANCE
```

We believe this to be quite simple and efficient. Moreover, a web-based interface will be available to interactively build such queries.

- Response time

Response time is a key issue when dealing with huge amount of data. No guarantee can be made besides best-effort. That said, network architecture and processing power can scale so that service outbound throughput is maximized (*c.f. § Scalability*).

- File formats

Regarding file formats, each user has its own preferences, depending on its habits, its research field or the software she uses. OPeNDAP allows several formats including DODS, HDF, NetCDF, ASCII and KML. As file generation is the last step in query response, and does not depend on how datasets are effectively stored, we have all freedom to implement any file formats, even streamable ones.

## Storage

As said in the introduction, annual data-growth rate is 100%. On the other hand, storage technologies follow Kryder's Law, that states that storage density grows at an annual rate of 40% [10]. That means that we can't expect solutions for storing petabytes from pure hardware improvements. Fortunately, very large scale storage solutions has emerged during past few years. Key-value stores, like CouchDB [12] or OpenStack/Swift [13] are able to efficiently store petabytes of data. They do so by sticking to a minimal semantic. Unlike distributed filesystems or relational databases, they avoid locks, have minimal access control management, or even are "eventually consistent".

How can key-value stores, also known as sharded databases, benefit to geo-referenced databases ?

First, we have to examine their limitations. Sharded databases are very efficient to store and retrieve relatively small objects (several megabytes, in general), but they will not store multi-gigabytes datasets without losing their competitive advantage over filesystems. Also, they do not comply with ACID model, as relational databases do. That is, operations can't be altogether Atomic, Consistent, Isolated and Durable.

Therefore, to ensure good storage scalability, we must enforce maximum stored object size, and live without guaranteed consistency.

- Keeping stored object size under control

A global parameter of the system is the optimal stored object size. From hundreds of kilobytes to several megabytes, depending on database and hardware used.

When adding a dataset to a database, import module first considers its parameters. Each parameter is stored in a different object in the store, with a simple naming scheme. For example "`/database_name/dataset_id/parameter_name`". If resulting object is larger than the optimal object size, it is recursively splitted along longitude and latitude, and then along parameter dimensions, until object has an appropriate size (figure 3).

Naming scheme is then like "`/database_name/dataset_id/parameter_name/tile_id`".

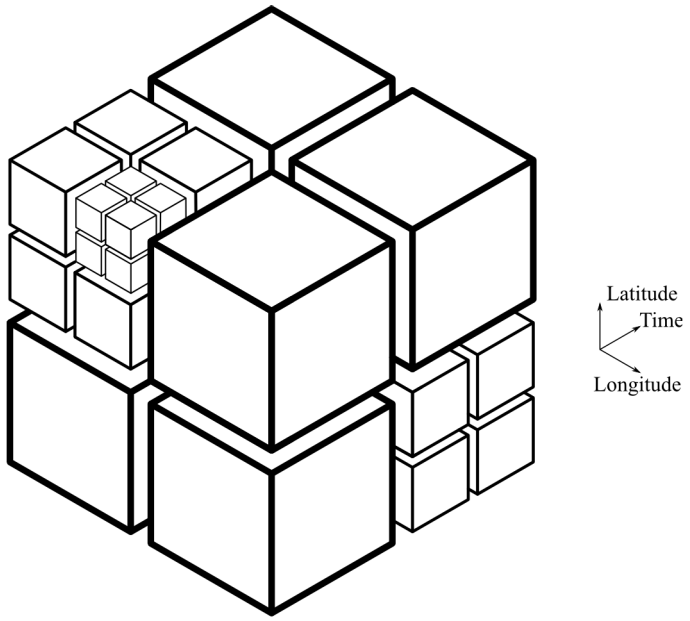


Figure 3: Three dimensional binary tiling, with "Time" parameter as third dimension.

Such a binary-tiling is not unlike QuadTiles as used in OpenStreetMap for images [14], or a generalized REG-REG tiling as implemented in ArrayStore [15].

Splitting along longitudes and latitudes is absolute. It does not depend on geographical extent of dataset, so that tile_id can be compared across databases. Moreover, a carefully chosen binary naming scheme

allows, along compactness, tile manipulations using only binary algebra, without the need of database lookup nor tree traversal, not even trigonometry (see figure 4).

With this kind of tiling, we can spread datasets among as many storage nodes as we want. This will be helpful when considering balanced computing (*c.f. § Scalability*).

- Atomicity and Isolation

Very much like in SciDB [16], we don't want a full ACID model, as this would require fully synchronous operations, that don't scale well.

Instead, a partially synchronous model is used, implemented on top of an asynchronous framework (twistedmatrix in our preliminary implementation). As shown in [17], we can build an atomic, isolated and durable scalable system, with some compromises on consistency. Care must be taken on how to do this. Particularly, adding a dataset must be an atomic operation. That is, metadata should hit the metadata database only when data has been successfully uploaded to the dataset storage. This implies that database and key-value store can be be isolated from each other except at at ingest time.

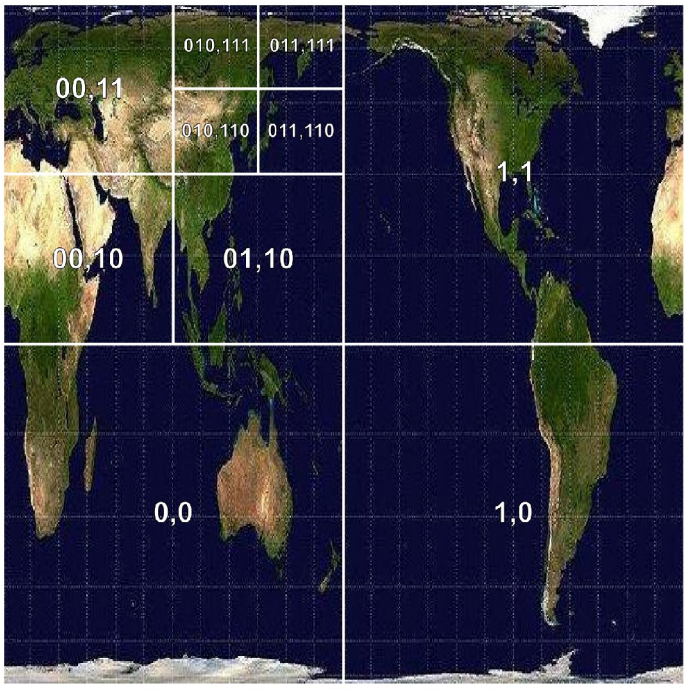


Figure 4: Binary tiling naming scheme

$(\bar{x},\bar{y})$ sub-tiles are $(\bar{x}0,\bar{y}0)$, $(\bar{x}1,\bar{y}0)$, $(\bar{x}1,\bar{y}1)$ and $(\bar{x}0,\bar{y}1)$

## Scalability

System scalability has software as well as hardware constraints. Carefully designing software and balancing hardware will optimize costs and performance ([18], [19]). At thousands of nodes scale, we must also consider hardware failure detection issue.

- Balanced computing

Storage scalability was exposed in previous paragraph. Aside from storage, computational power and data throughputs should scale as well. Geosciences have, indeed, data-intensive workloads. That is, processor operations need data. High data-intensive workload is defined as having very low floating point operations (between 0 and 10) per byte of input data [4]. GPU-computing epiphany increased even more pressure on data paths.

As storage is already spread among a huge number of nodes, these nodes are also used to do server-side computations, thus aggregating data paths (disk to memory, memory to cpu, cpu to network and node to node). Balanced with network throughputs, we have large freedom in hardware choices.

Moreover, considering that "electricity costs (will) soon exceed the purchase price of the computing hardware" [20], power efficiency of the solution must also be considered. Having very parallel processes and data flows alleviate the need of power-dissipating hardware, and low-power components can be used almost everywhere in the storage cluster.

Reference installation is running on several servers. Each one is made of low-power dual-core CPU, with three "green" SATA disks attached (aggregated throughput [21] of 300MB/s), DDR3-800 SDRAM (6.4GB/s) and 1Gb/s network (125MB/s), all through PCI-Express 2.0 links (2GB/s), for an estimated power of about 50W. For WMS map generations, as seen in [22], outbound 1Gb/s link is saturated as soon as we have two nodes in storage cluster, while in-cluster network throughputs are balanced between nodes. Further experiments will show how far this will linearly scale, but we believe that we could output WMS queries at 10Gb/s from 30TB of data with less than 20 nodes and less than 1kW of electrical power.

- Byzantine processes

Another aspect of scalability is undetected errors. On a cluster of thousands of nodes, hardware failure will occur. Most components have failure detection (S.M.A.R.T. and RAID for disks, ECC for memories, checksums for network packets, etc...), but we need more global monitoring to detect and isolate byzantine nodes.

Solution is to benefit from storage redundancy. For example, OpenStack/swift requires data to be stored on at least three servers (and advise against using RAID for disks). Noting that dataset extractions are deterministic, orchestrator can ask several nodes to perform the same task, and compare results. This lead to twice the computations, but not necessarily twice the network: a checksum-hash of the result can be asked to one node, and, on the other hand, computed with data received from another. A standard ballot mitigation is then used to detect which node shows byzantine behavior.

## Server-side computations

As describe in [4], clustered server-side computations will benefit of locality of data and parallelism, but comes with its own set of problems. While some operations are by nature "embarassingly parallel" (*e.g.* image generation), others need algebraic work to perform in parallel (*e.g.* NCO).

- Image generations

A WMS request barely only builds images. Implemented OPeNDAP server already distributes workload among nodes to build tile-based sub-images, that are then aggregated in the orchestrator layer's cache and finally compressed in the protocol layer to wanted image file format.

- NetCDF Operators

NCO, NetCDF Operators, is a set of operators commonly used in geosciences to perform arithmetic operations and data permutation on datasets [23]. With pragmatic design in mind, it defines the set of operations that we need to implement in DfP. Parallelization of such operators is not trivial, and we are planning SWAMP [4] integration to achieve this.

## Long-term preservation

Datasets, as uploaded by producers, must be saved on the long run. This is the "Preservation planning" part in the OAIS model (figure 2).

They don't obey the same access schema as on-line, operational ones. We believe that there will be far less downloads of these original datasets. Because they are in random formats (HDF, spreadsheet, GRIBS, proprietary, etc...), may be hard to interpret (random geodesic reference system, time-base,

etc...), and even hard to find. Nonetheless, some may find useful to be able to get them. Also, we must keep them for reference, future reprocessing, and, more importantly preservation.

Preservation, that is not simple backup, has three main objectives [24]: keep the document, preserve intelligibility and make it accessible.

Keeping the datasets does not have the performance constraint that we have for extractions, computations and operational use. We don't necessary have to shard files and maintain some home-brewed reconstruction software. In fact, we instead have to use a standard, open and reliable storage. Distributed file systems (AFS, ZFS, GFS, and many others) are solutions of choice. A large user base ensures that, in a foreseeable future, we will be able to find manpower to retrieve these data. Similarly, open standards and non-commercial-closed solutions ensure that we will have software to do so.

Preserving intelligibility means data will be understandable even if metadata database is lost or inaccessible. Briefly put, we have to store data and its metadata together. A simple human-readable meta-metamodel is used for metadata format description (in XML or JSON, for example).

Retrieving filesystem-based archives is a matter of Universal Resource Location (URL) naming scheme, and some basic web-based portal to access these. Proposed server stores archived datasets URLs as part of datasets metadata, and make them accessible via the same web portal. At ingest time, a zipped archive is generated with data and metadata and is made available at an URL of the form "`/orig/database_name/dataset_id.zip`".

## CONCLUSION

Most data portals only consider metadata, leaving data publication and conservation responsibilities to producers. Moreover, current OPeNDAP implementations have hard time when dealing with petabytes of data, making data exploitation cumbersome if not haphazard.

This paper described general data-flow and scalability for a petabyte-scale durable data warehouse.

Preliminary implementation of described OPeNDAP server has permitted us to validate feasibility and to tackle with real-world problems [5]. Still, more work has to be done for reaching a production-grade service.

## REFERENCES


[1] - "Lessons Learned from Managing a Petabyte", Jacek Becla *et al.*, Babar and Babar Computing Group Collaborations, SLAC-PUB-10963 (2005), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.2085&rep=rep1&type=pdf

[2] - "Reference Model for an Open Archival Information System", CCSDS Blue Book (2002), ISO14721:2003, http://public.ccsds.org/publications/archive/650x0b1.pdf

[3] - "Reference Model for an Open Archival Information System", CCSDS Pink Book (2009), http://public.ccsds.org/sites/cwe/rids/Lists/CCSDS 6500P11/Attachments/650x0p11.pdf

[4] - "Efficient Clustered Server-side Data Analysis Workflows using SWAMP", Wang, D. L., C. S. Zender, and S. F. Jenks (2009), Earth Sci. Inform., 2(3), 141–155, doi:10.1007/s12145-009-0021-z, http://dust.ess.uci.edu/ppr/ppr_WZJ091.pdf

[5] - "Googleearth as PyCTOH client", G. Pernot, (2010), http://www.youtube.com/watch?v=YQmaKMC6q2Y

[6] - "Mixing structures in the Mediterranean sea from Finite-Size Lyapunov Exponents", F. d'Ovidio, C. López, E. Hernández-García, V. Fernández, Geophys. Res. Lett., 31, L17203 (2004)

[7] - "Stirring of the Northeast Atlantic spring bloom: a Lagrangian analysis based on multi-satellite data", Y. Lehahn, F. d'Ovidio, M. Levy, E. Heifetz, J. Geophys. Res., 112, C08005 (2007)

[8] - "The Data Access Protocol – DAP 2.0", Gallagher, Potter, Sgouros, Hankin, Flierl (2011), http://www.esdswg.org/spg/rfc/ese-rfc-004/ESE-RFC-004v1.2.pdf

[9] - "PyCTOH documentation", G. Pernot (2010), http://pyctoh.nongnu.org/index.html

[10] - “After Hard Drives - What Comes Next?”, Kryder and Chang Soo Kim. IEEE Transactions on Magnetics, Vol. 45, No. 10, (2009), doi:10.1109/TMAG.2009.2024163

[11] - “MODIS Snow Cover Product”, Aqua/AMSR-E and Modis Data Product Handbook, vol. 2 , pp 211-212, http://modis.gsfc.nasa.gov/data/dataprod/pdf/MOD_10.pdf

[12] - “CouchDB – The definitive guide : Time to relax”, J. Chris Anderson, Jan Lehnardt and Noah Slater, O'Reilly Media (2010)

[13] “OpenStack Object Storage Administrator Manual”, OpenStack LLC (2011), http://docs.openstack.org/cactus/openstack-object-storage/admin/os-objectstorage-adminguide-cactus.pdf

[14] - “QuadTiles”, OpenStreetMap contributors (2010), http://wiki.openstreetmap.org/wiki/QuadTiles

[15] - “ArrayStore: A Storage Manager for Complex Parallel Array Processing”, Emad Soroush, Magdalena Balazinska, and Daniel Wang. SIGMOD'11, June 12-16, 2011, Athens, Greece, http://www.scidb.org/Documents/SciDB-sigmod362-soroush.pdf

[16] - “Overview of SciDB – Large Scale Array Storage, Processing and Analysis”, The SciDB Development Team, SIGMOD'10, http://www.scidb.org

[17] - “Brewer's Conjecture and the Feasibility of Consistent Available Partition-Tolerant Web Services”, Seth Gilbert and Nancy Lynch, in ACM SIGACT News (2002), http://theory.lcs.mit.edu/tds/papers/Gilbert/Brewer6.ps

[18] - “Petascale Computational Systems: Balanced CyberInfrastructure in a Data-Centric World”, Gordon Bell, Jim Gray and Alex Szalay (2005), http://arxiv.org/pdf/cs.DB/0701165

[19] - “InfiniBand, PCI Express, and Intel® Xeon™ Processors”, Mellanox Technologies, http://www.mellanox.com/pdf/whitepapers/Balanced_Computing_WP_040.pdf

[20] - “Report from the 3rd Workshop on Extremely Large Databases”, Jacek Becla, Kian-Tat Lim and Daniel Liwei Wang, Data Science Journal, Volume 8, pp MR1-MR16 (2010), http://www.jstage.jst.go.jp/article/dsj/8/0/MR1/_pdf

[21] - “List of device bit rates”, http://en.wikipedia.org/wiki/List_of_device_bandwidths

[22] - “Finite-Size Lyapunov Exponents from Altimetry”, G. Pernot (2010), http://www.youtube.com/watch?v=wUqQGEkMsyU

[23] - “Analysis of Self-describing Gridded Geoscience Data with netCDF Operators (NCO)”, Zender, C. S. (2008), Environ. Modell. Softw., 23(10), 1338–1342, doi:10.1016/j.envsoft.2008.03.004, http://dust.ess.uci.edu/ppr/ppr_Zen08.pdf

[24] - “Digital long term preservation vs. long term secure backup”, CINES (2009), http://www.cines.fr/spip.php?rubrique229&lang=en