Geosci. Model Dev. Discuss., 8, C2821–C2825, 2015
www.geosci-model-dev-discuss.net/8/C2821/2015/

Geoscientific Model Development Discussions

Interactive comment on "Distributed visualization of gridded geophysical data: a web API for carbon flux" *by* K. A. Endsley and M. G. Billmire

K. A. Endsley and M. G. Billmire

endsley@umich.edu

Received and published: 7 November 2015

Thanks for your feedback! We'll address your comments one at a time below.

> "Authors claim CDE is offering distributed visualization, however, this is not substantiated in detail; from what I can infer data are always loaded from (server) local files. Visualization is addressing 3D x/y/t cubes plus multi-variables, but it remains essentially 2D plus "movie", no 3D portrayal is mentioned."

The data cube metaphor is not meant to imply 3D visualization; rather, three to four variables–planar coordinates, measurement value, and time–are represented. Altitude

is clearly an important dimension in climate datasets but the CDE does not support a joint visualization of measurement value and altitude.

> "[W]hat is the exact storage scheme for the datacube in MongoDB?"

Data cube(s) are stored as one or more "scenarios" which may each correspond to a single model run (with particular parameters) or multiple model runs under some unified conditions. Scenarios might also separately encapsulate measurement values and uncertainty, allowing them to be viewed side-by-side in the Coordinated View subsystem. Page 5748, Lines 2-4 state: "Each scenario has one timeline associated with it and gridded data belonging to that scenario are uniquely keyed by their date and time." Slices in time ("X-Y" slices) of the data are stored. Page 5747, Lines 18-19 state: "Specifically, the data are stored and transmitted as JavaScript Object Notation (JSON) documents." We will revise the quoted lines to make this more clear.

> "[W]hat part of the analysis is pushed into MongoDB, and what is solved in the middleware?"

Page 5751, Lines 13 through 18 read: "The temporal aggregation is handled by the MongoDB aggregation pipeline, which facilitates very fast aggregation of multiple X–Y slices (maps spanning time). Spatial aggregation of one or more pixels (an aggregate value spanning a spatially filtered subset) is achieved using a combination of the JSTS Topology Suite JavaScript library and MongoDB's geospatial query operators." We will revise and extend this to elaborate that temporal aggregation *and differencing* are handled by the MongoDB aggregation pipeline; that calculation and display of anomalies is done client-side in JavaScript; that population summary statistics are calculated in the Python API and stored in MongoDB; that all other visualization tweaks and statistical stretching is done "on-the-fly" in JavaScript.

> “In how far do the authors consider this architecture scalable? To this end, at least a few performance figures would have been helpful, even better so a comprehensive evaluation: what are response times in general? where in the architecture is time spent, eg: how much of the query response time goes into MongoDB, and how much into the JavaScript middleware? How does this compare to, eg, C/C++ implementations?”

We will provide performance metrics in the revised manuscript.

> “[W]hile the paper mentions some tools and one standard (WMS) in the field it lacks a solid comparison against immediately “competitors”. Hadoop, Array Databases, as well as virtual globes like NASA WorldWind come to my mind.”

We will provide more context for the CDE in relation to Hadoop, Array Databases, and NASA WorldWind within the revised manuscript.

> “[D]ata ingestion, ie: massaging heterogeneous incoming data to a suitable service structure, typically is an involved task. Section 2.2 does not detail on this, which would be interesting to know: what challenges had to be met? Any innovative approach taken?”

Some challenges may be merely mundane details for some readers... We discovered that Matlab has changed the format of its saved binary output files over the years from an HDF4-like structure to one requiring an HDF5-compatible reader. We selected Python and its essential SciPy library as together they provide support for both HDF4 and HDF5 formats. Thus, our experience is a reminder of the importance of backwards compatibility, which is likely well-recognized in the scientific programming community.

We will add a brief remark on this and other considerations, such as the need for calculating population summary statistics offline, at the end of Section 2.3.

> “[M]assive binary data encoded in text form seems like a big impediment for transfer and processing. MongoDB querying certainly does not offer competitive performance on datacubes, and only limited functionality. Unfortunately, the paper remains superficial here and does not explain the detailed storage schema.”

Our experience with the CDE is that, for regional and global gridded climate datasets, there are no significant impediments to display and analysis on the web. Comparisons of the performance of NoSQL databases lacked consensus at the time we began development (in 2012) but have since begun to show that, indeed, Cassandra and HBase provide better performance in most applications than MongoDB (e.g., Dede et al. 2013 in *Proceedings of the 4th ACM Workshop on Scientific Cloud Computing*). However, for single nodes, MongoDB can be still provide equal or better performance than alternatives such as Hadoop (as cited by Nyati et al. 2013, at *International Conference on Advances in Computing, Communications and Informatics, ICACCI*). As for its “limited functionality,” in working with our domain experts, the key analytical capabilities they needed were all implemented in the CDE using either MongoDB or more practical front-end capabilities. In presenting early versions of the software to groups like DataONE and the OCO-2 Science Team, we did not identify any analytical workflows for these types of data (Level III gridded products) that we could not support.

> “Example in 2.4: the result to me, following the query logic, should be a 3D cube extending allong the full spatial footprint and temporally reduced to the start and end point indicated in the query. However, authors call the result a ‘timeseries’ which earlier has been introduced as being 1-D. This might be clarified.”

Only 1D time series or 2D slices of the data cube are made available through the web API. The “t.json” endpoint is not constrained in its design to deliver 1D time series, however, as 2D time series were not required for any of the features identified by the user community and would be expensive to generate, the “t.json” endpoint requires “aggregate” and “interval” keywords so that it can deliver a time series. We will expound on this in the revised manuscript.

> “[A]uthors characterize retrieval from MongoDB as ‘very fast’ but without indicating measurements, and no comparison to tools offering the same functionality.”

Performance metrics for the CDE will be included in the revised manuscript.

Interactive comment on Geosci. Model Dev. Discuss., 8, 5741, 2015.