Geoscientific Model Development Discussions

Geosci. Model Dev. Discuss., referee comment RC1
https://doi.org/10.5194/gmd-2021-138-RC1, 2021

Comment on gmd-2021-138

Michael Kuhn (Referee)

Referee comment on "A Parquet Cube alternative to store gridded data for data analytics and modeling" by Jean-Michel Zigna et al., Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2021-138-RC1, 2021

The authors propose to use Parquet for storing gridded data efficiently. They compare NetCDF (using the THREDDS Data Server), NetCDF on S3, Pangeo-Zarr and Spark-Parquet using a number of different scenarios.

The comparison shown in the paper is interesting, but suffers from a few flaws: It is hard to compare the different results since the experiments use different hardware configurations and different compression algorithms. However, most notably, Pangeo-Zarr seems to perform best in majority of experiments (sometimes being orders of magnitude faster than Spark-Parquet) but the authors still propose using Spark-Parquet. Overall, the benefits of Spark-Parquet do not become clear. The paper would benefit significantly from more thorough explanations and uniform experiment configurations.

General comments and questions:
- 41: The citation format doesn't seem to adhere to the journal's guidelines (https://www.geoscientific-model-development.net/submission.html#references). This also applies to the following citations.
- Figure 1: All figures should be referenced explicitly in the text. This also applies to all other figures.
- 146: Why do you describe the system configuration here? It's also mentioned later.
- 182: The parallel processing scenario is not very detailed. How does it work? What happens in parallel? In general, all three scenarios could use some more explanations, for instance, showing their (pseudo) code.
- 192: The code for the different benchmarks should be provided to allow readers to compare/reproduce them. The Git repository only seems to contain code for converting NetCDF to Parquet.
- 193: How is the THREDDS implementation used, a Java application or something else? It does not become clear from the description.
- 229: Pangeo-Zarr uses LZ4 compression. Does NetCDF also use compression in your experiments? If not, Compression can add significant overhead, how are you able to compare them fairly?
- 250: How does this scalable process work?
- Figure 3: What does this figure show? It does not become clear from the text. Moreover,

the text in the figure is quite blurry and its resolution should be increased.
- 301: Spark-Parquet seems to use Snappy, which has different performance characteristics than LZ4 used previously. For a fair comparison, all approaches should use the same compression algorithm (or none).
- Figure 5: This figure shows that Snappy and LZ4 result in different file sizes, which in turn means that the benchmarks will have to read/write different amounts of data, possibly skewing the comparison.
- 330: Every scenario seems to run on different hardware, making it hard to compare the results. 512 GB vs. 12 GB RAM will have significant impact on caching. Pangeo-Zarr uses GPFS, while NetCDF is stored on local disks. GPFS is expected to be much faster than a local disk.
- 350: How did you make sure they were using the same amount of memory? Did you also account for differences in CPU performance (apart from using the same amount of cores)?
- Table 2: Why is THREDDS-S3 larger than NetCDF? Aren't they both using NetCDF?
- 373: What are these numbers supposed to tell the reader?
- 387: Pangeo-Zarr and Spark-Parquet were also using NetCDF then? According to Table 2, all of their datasets were below 1 TB.
- 395: What does "10 cores of 4 Gb" mean? 4 GB in total or 4 GB per core?
- Figure 9: The different scaling on these figures implies that both Pangeo-Zarr and Spark-Parquet had similar performance but Pangeo-Zarr was significantly faster.
- Figure 11: The different scaling makes it hard to compare results again.
- Figures 15 and 16: These figures seem to imply that Pangeo-Zarr has much better scaling behavior than Spark-Parquet. Can this be explained somehow?
- Figure 18: Why are you only using 1 to 3 cores here?
- 534: Why do you compare 50 to 40 cores? That makes it hard to judge the differences.
- 539: It seems NetCDF-S3 has several limitations. Have you also considered HDF5 (https://www.hdfgroup.org/solutions/enterprise-support/cloud-amazon-s3-storage-hdf5-connector/)?
- 555: You mention that Pangeo-Zarr is limited to Python. Isn't THREDDS also limited to Java? Which languages are supported by Spark-Parquet?
- 586: How do you support this conclusion? Pangeo-Zarr seems to be the fastest format and Python most likely has the largest community. Moreover, Pangeo-Zarr shows much better scaling than Spark-Parquet in a few of your experiments.
- 615: The last access dates for all references seem to be in French.

Layout problems, typos etc.:
- 186: "4 go" - Is this supposed to be "4 GB"?
- 209: "manor" - Should be "manner".
- Figure 4: This figure is also blurry.
- Figures 5 and 6: The axis descriptions are very hard to read due to the low resolution.
- 339: "Go" - Should be "GB". Also applies to the following lines.
- Figure 11: The second row seems to be titled incorrectly, THREDDS-NC is missing and instead Spark-Parquet is shown twice.
- Figure 19: This figure is so blurry that it's impossible to read.