Use of Schema on Read in Earth Science Data Archives

IN31A-0068

NASA/Goddard EARTH SCIENCES DATA and INFORMATION SERVICES CENTER (GES DISC)

Mahabaleshwara Hegde[1,2], Christine Smit[1,4], Paul Pilone[3], Maksym Petrenko[1,2], Long Pham[1]

[1] NASA Goddard Space Flight Center, [2] ADNET Systems Inc., [3] Element84, [4] Telophase Corporation

Problem – Querying Big Data

- Geospatial data are traditionally stored in files organized along time dimension (e.g., as daily or hourly ‘granules’)
- File formats and structure vary from mission to mission, or even within the same mission
- These factors present an obstacle to quick adoption of new datasets and make it hard to query big data in dimensions other than time

Solution – Parquet and Frameworks

- Modern column-oriented data storage formats provide an efficient framework for storying large-scale data in a format optimized for querying.
- Apache’s **Parquet** is a popular column-oriented data format supported in multiple frameworks:
  - Scala, PySpark, Pandas, sparkR, etc.
- We evaluate Parquet at GES DISC as a next-generation Earth data storage format in a prototype Cloud-based Giovanni (see poster **IN23A-0080** for more details on Giovanni).

Key Features of Parquet

+ Support for very large data files
+ Compressible - comparable to compressed HDF5
+ Can add new data and columns to existing files
+ Efficient querying with declarative SQL-like language (e.g., spark, pandas)
+ Optimization for chained queries & parallelization
+ Multiple files can be queried as a single file
+ Query an entire data set as a single database table!
+ Cloud-friendly - can read parquet from S3 buckets
+ Built-in metadata
+ Files are self-describing and ‘instantly’ usable
+ Can be used in schema-on-read applications
- Certain limitations to the metadata and data structures – not a full replacement for archive data

Big Data Queries Made Simple

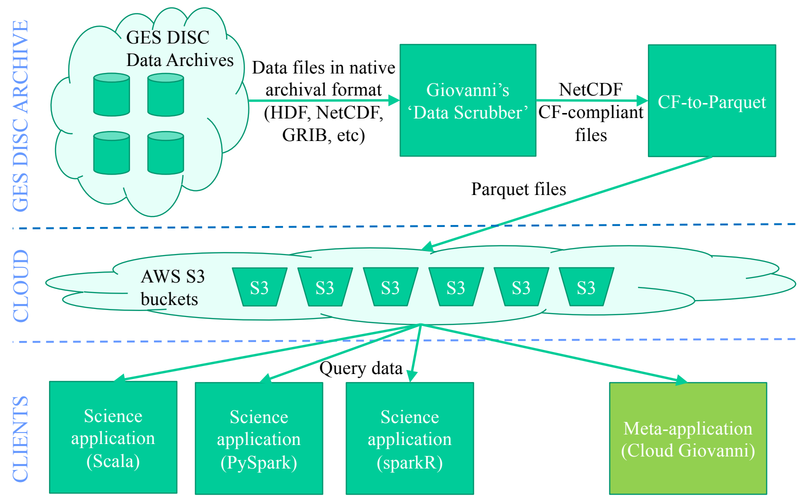
Sample Scala code for computing daily weighted averages of ozone over 10°N,10°W,10°S,10°E area for entire dataset (Level 3 daily 0.25°x 0.25° OMTO3e O3)

```
sqlCtx.read.parquet("parquetFile").createOrReplaceTempView(”dataset")
dataAndWeightsCtx = sqlCtx.sql("SELECT time, Ozone, COS(RADIANS(lat)),
lat, lon as weights FROM dataset
   WHERE lat>=-10 AND lat<=10 AND lon>=-10 AND lon<=10.0")
dataAndWeightsCtx.createOrReplaceTempView("dataAndWeights")
timeSeriesCtx = sqlContext.sql("SELECT time,
    SUM(Ozone * weights) / SUM(weights) as var_average
    FROM dataAndWeights GROUP BY time ORDER BY time")
timeSeriesCtx.createOrReplaceTempView("averages")
sqlCtx.sql("SELECT SUM(var_average) FROM averages").show()
```

Sample OMTO3e Parquet File

row	lat	lon	time	Ozone
0	-87.5	-179.5	1096588800	143.000000
1	-87.5	-178.5	1099180800	142.600006
…	…	…	…	…
1753009	70.5	178.5	1099180800	395.100006
1753010	70.5	179.5	1099180800	391.299988

Parquet in the Cloud


GES DISC ARCHIVE
GES DISC Data Archives
Data files in native archival format (HDF, NetCDF, GRIB, etc)
Giovanni’s ‘Data Scrubber’
NetCDF CF-compliant files
CF-to-Parquet
Parquet files
CLOUD
AWS S3 buckets
S3
S3
S3
S3
S3
S3
Query data
CLIENTS
Science application (Scala)
Science application (PySpark)
Science application (sparkR)
Meta-application (Cloud Giovanni)

See also

IN23A-0080: Giovanni in the Cloud: Earth Science Data Exploration in Amazon Web Services
IN41B-0039: The Value of Data and Metadata Standardization for Interoperability in Giovanni

Performance Analysis

- We computed the daily cosine average of daily 0.25° L3 OMTO3e O3 over 10°N,10°W,10°S,10°E box during 2004-10-01 through 2017-06-08.
- 4540 original daily OMTO3e granules were converted into Parquet as either a single file, 14 yearly files, 151 monthly files, or 4540 individual daily files.
- Parquet files were stored either locally or on AWS S3.
- Computation was performed with 1 or 2 threads.
- Parquet code (see on the left) also summed up all averages to force ‘reduce’ operation (i.e., collect data back from all threads)
- Results are from the 1st run. Subsequent runs from AWS were as much as 2x faster due to caching (not shown).

Observations

Files	NetCDF convert to Parquet (hh:mm)	Parquet Packaging	Time (mm:ss)			
			Local files		AWS S3	
		Threads:	1	2	1	2
4540	--	Baseline - Giovanni	23:00	--	--	--
1	12:40	Single Parquet file	00:34	00:26	04:30	01:40
14	02:20	Packaged by year	00:33	00:24	06:00	05:26
151	01:40	Packaged by month	00:37	00:26	04:34	02:31
4540	01:30	Packaged by granule	01:33	01:08	35:16	18:31

- Packaging data into single files provides the best performance, but is more expensive to generate and update, and has the smallest boost from caching on repeat queries when used with AWS S3.
- Packaging by month perform comparable to single files, but is cheaper to manage and works better with network caching.
- Using multiple threads provides a noticeable increase in performance, especially if the bulk of computations can be completed on compute nodes before a reduce operation.