RECHUNKER: THE MISSING LINK FOR CLOUD OPTIMIZED DATA

RYAN ABERNATHEY & TOM AUGSPURGER
OPEN CLOUD ARCHITECTURE

Analysis Ready Data
Cloud Optimized Formats

Scalable Parallel
Computing Frameworks

data provider's $ightarrow$

cloud region

Data Provider's $ightarrow$

compute cluster

Data Consumer's $ightarrow$

figures

- Parquet
- COG
- [tile]DB
- Zarr

- Apache Spark
- DASK

- Google Cloud/Regional
- AWS
- Azure
- Jetstream
Chunked Array Formats

- Break up large arrays into smaller, more-manageable sub-arrays: “chunks”
- Chunks define a unit for parallelism
- Play well with distributed computing frameworks (e.g. Dask)
PROBLEM: CHUNKS NOT ALIGNED WITH ANALYSIS

✅ Calculate global statistics at each point in time

❌ Calculate timeseries statistics as each point in space
PROBLEM: CHUNKS NOT ALIGNED WITH ANALYSIS

Best practices to go from 1000s of netcdf files to analyses on a HPC cluster? 📄

What?
8759 netcdf files totalling 17TB of HYCOM ocean model (u,v) velocity data at two depth levels (and bottom velocity), at hourly time steps. So the total data arrays of interest are [9000 (X) by 7055 (Y) by 8759 (time)] for both u and v. Using nco tools, I can reduce these files as an example to 365 netcdf files, totaling 4.3TB, one file per day (24 hourly steps), for one depth level only, so data arrays are 9000 (X) by 7055 (Y) by 8759 (time) for the u,v components.

35 posts later...
LAZY RECHUNKING (DASK)

FROM

Array | Chunk
---|---
Bytes | 512.00 MB | 128.00 MB
Shape | (4, 4000, 4000) | (1, 4000, 4000)
Count | 4 Tasks | 4 Chunks
Type | float64 | numpy.ndarray

TO

Array | Chunk
---|---
Bytes | 524.29 GB | 125.96 MB
Shape | (4096, 4000, 4000) | (4096, 62, 62)
Count | 258963 Tasks | 4225 Chunks
Type | float64 | numpy.ndarray

- Requires global communication
- Possibly uses lots of memory
- Possibly creates a huge number of tasks
Rechunker is a Python package which enables efficient and scalable manipulation of the chunk structure of chunked array formats such as Zarr and TileDB. Rechunker takes an input array (or group of arrays) stored in a persistent storage device (such as a filesystem or a cloud storage bucket) and writes out an array (or group of arrays) with the same data, but different chunking scheme, to a new location.

Rechunker is designed to be used within a parallel execution framework such as Dask.

See the documentation for more.
Design Principles

- **Respect memory limits.** Rechunker’s algorithm guarantees that worker processes will not exceed a user-specified memory threshold.

- **Minimize the number of required tasks.** Specifically, for $N$ source chunks and $M$ target chunks, the number of tasks is always less than $N + M$.

- **Be embarassingly parallel.** The task graph should be as simple as possible, to make it easy to execute using different task scheduling frameworks. This also means avoiding write locks, which are complex to manage.
ALGORITHMS

Push

- Read each source chunk only once
- Write source data to many target chunks
- Problem: need to synchronize writes

Pull

- Write each target only once
- Read each source file many times
- Problem: can blow out memory, serial loop over source chunks may be very slow
ALGORITHMS

Push / Pull
ALGORITHMS

Push / Pull Consolidated
Open a Zarr Array

```python
group = zarr.open_consolidated(source_store, mode='r')
source_array = group['sla']
source_array
```

<zarr.core.Array '/sla' (8901, 720, 1440)

```python
source_array.chunks
```

(5, 720, 1440)

Make a Rechunking Plan

```python
max_mem = '1GB'
target_chunks = (8901, 72, 72)
store_tmp = gcs.get_mapper('pangeo-scratch/rabernat/rechunker_demo/temp.zarr')
store_target = gcs.get_mapper('pangeo-scratch/rabernat/rechunker_demo/target.zarr')
r = rechunker.rechunk(source_array, target_chunks, max_mem, store_target, store_tmp)
```

Rechunked

- Source
- Intermediate
- Target

Execute the Plan

```python
result = r.execute()
result
```

<zarr.core.Array (8901, 720, 1440) float64>
DASK GRAPH

Read Source + Write Intermediate

Read Intermediate + Write Target
ROADMAP

https://github.com/pangeo-data/rechunker
https://rechunker.readthedocs.io/

- Support other chunked array formats (e.g. TileDB)
- Support other execution frameworks (e.g. lambda)
- Command line interface
- Please get involved!