Leveraging DAOS storage system for seismic data storage

DUG'21

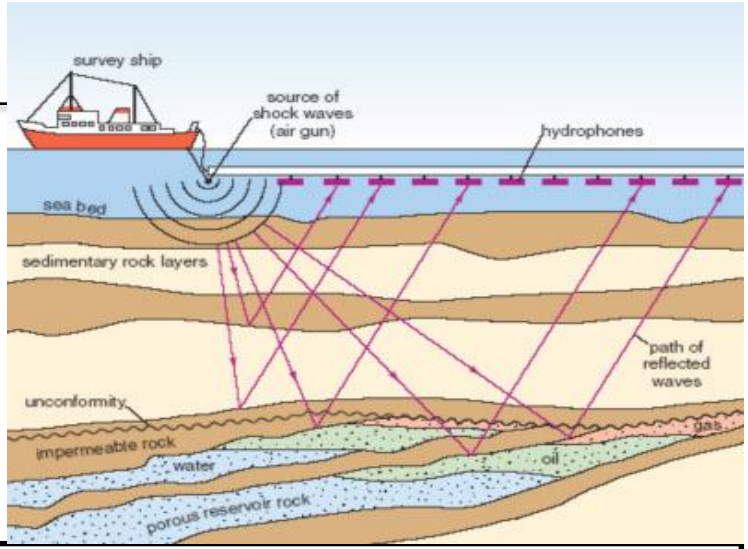
Omar Marzouk *, Mirna Moawad, Amr Nasr (Brightskies)

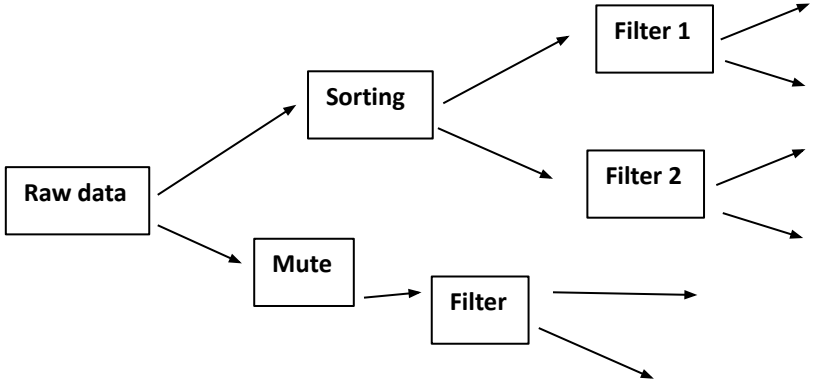
Johann Lombardi, Mohamad Chaarawi, Philippe Thierry , Michael Hennecke(Intel)

Sigrun Eggerling (Lenovo)

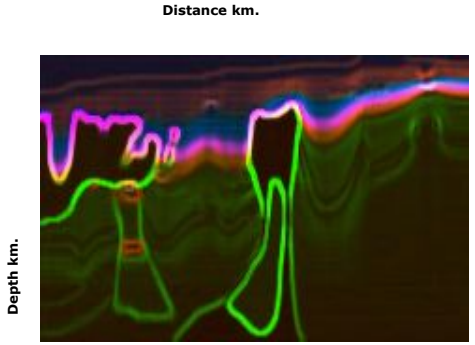
Marine seismic acquisition

- 1 shot :
  - 8 streamers*1000 receivers*5001 samples*4 bytes= 0.15 GB
- 1 line :
  - 20km / 25m = >> 800 shots per line = 120GB per line
- 2400 lines
  - **=> 280 TB of Raw seismic data** for about 1200 km² acquisition



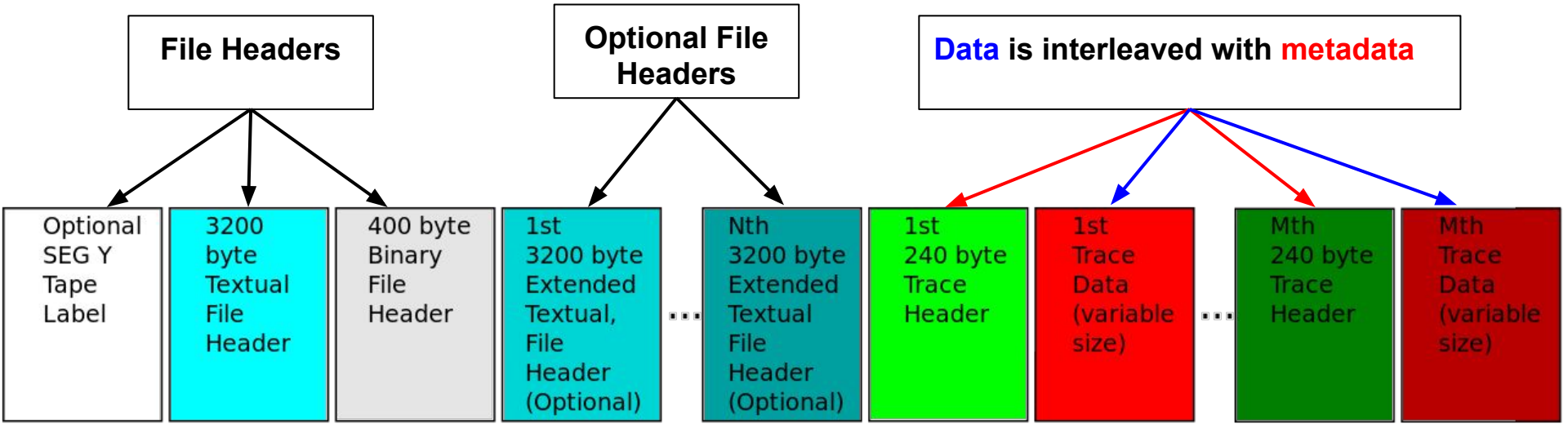


… to the final result



Seg-Y File Structure

Average size of raw seismic data obtained from a survey area of 1200 km^2 would be around 280 TB



Motivation

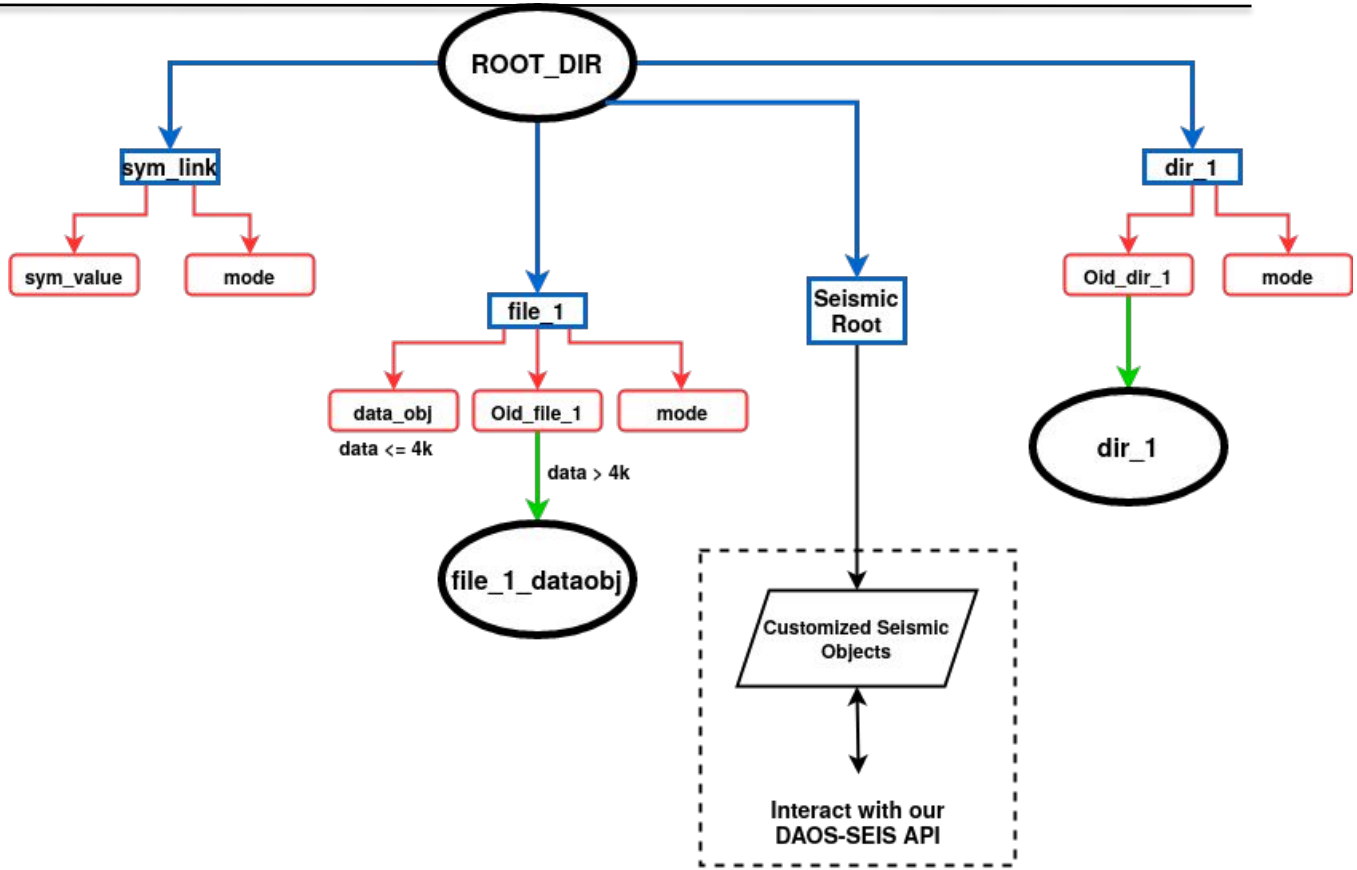
Limitations

- A new copy of the data is created along with each processing step.
- Serial accessing of data in a segy file.
- Most processing applications access traces headers to decide the access pattern to the data.

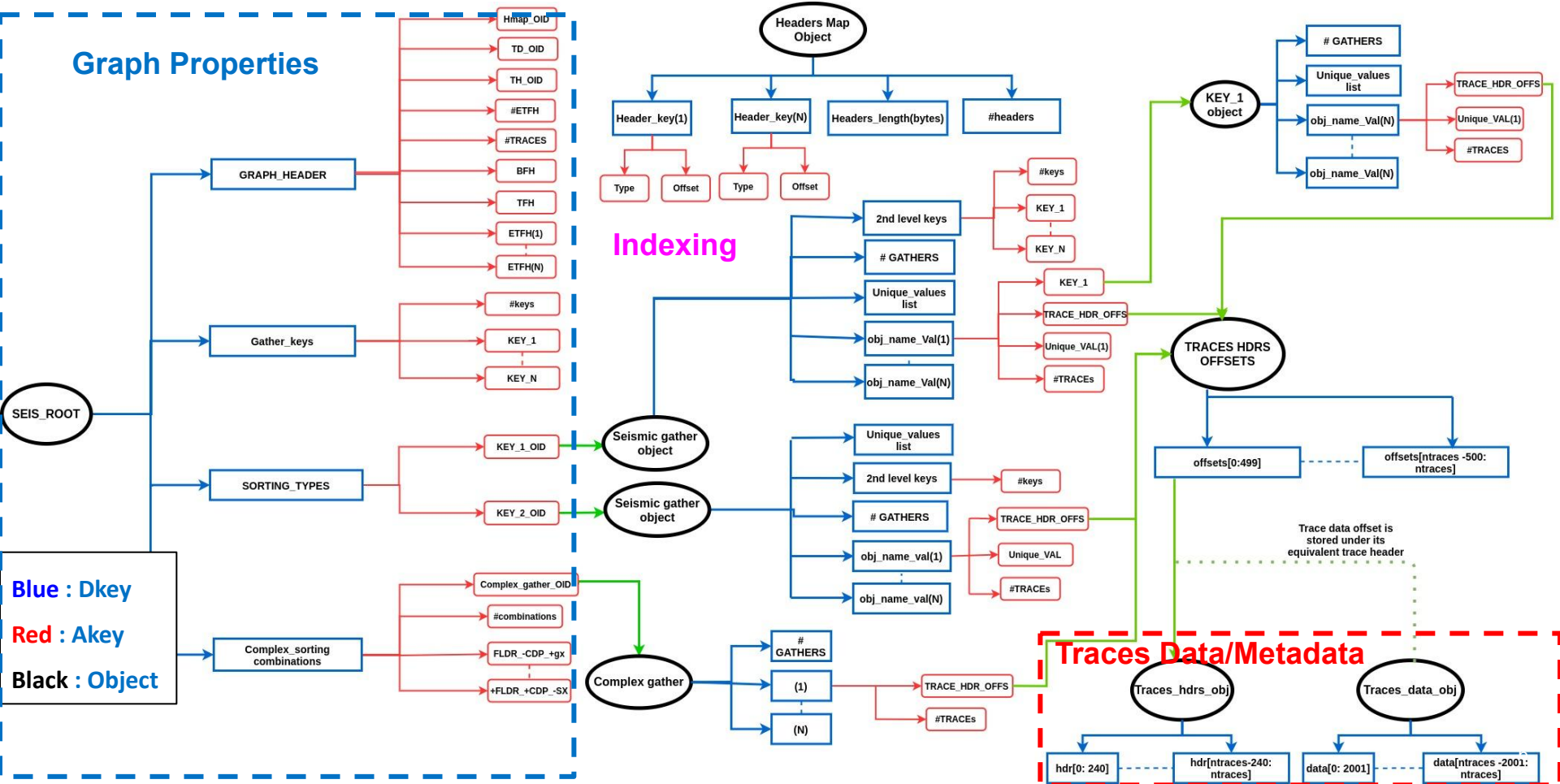
Goals

- Leverage object based storage system and inherent metadata/data separation.
- Reduce number of copies through utilizing daos snapshots.
- Store only updates through utilizing daos versioning object storage.

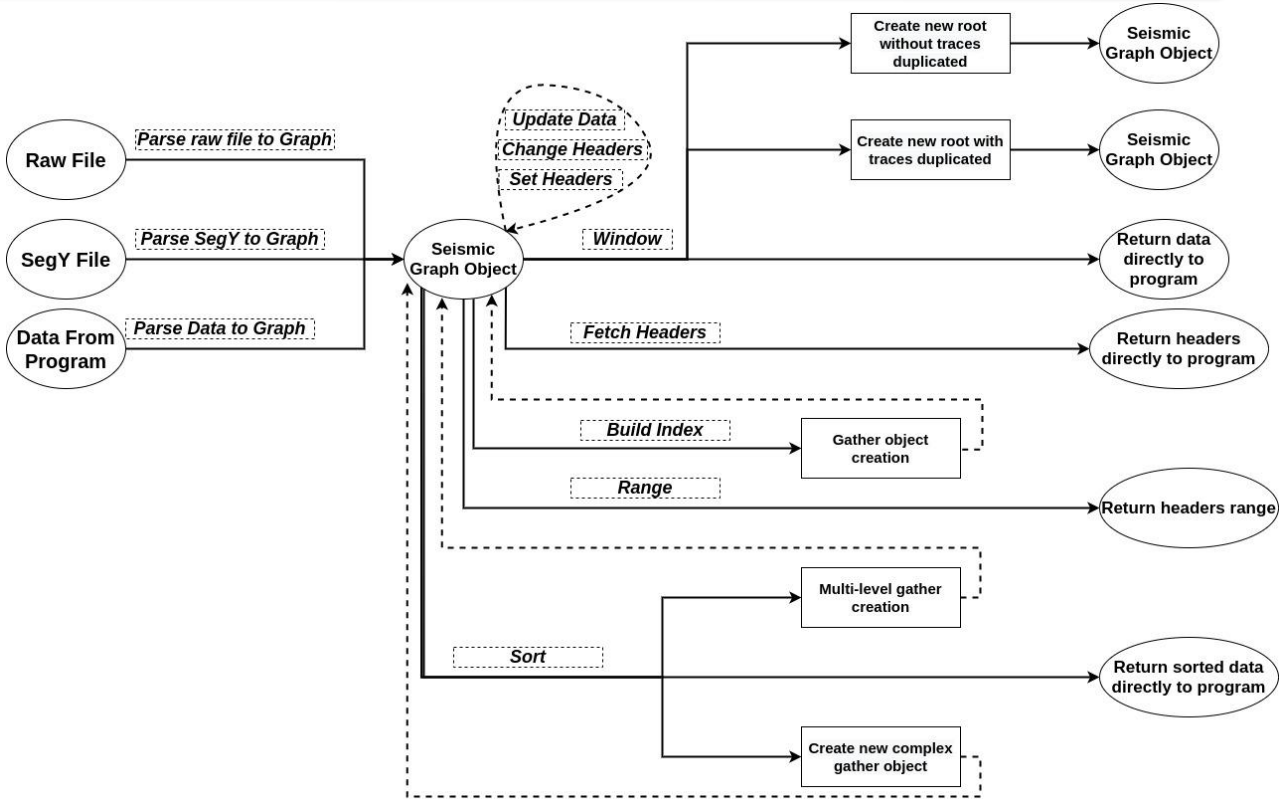
DAOS SEISMIC GRAPH (DSG)



DAOS SEISMIC GRAPH (DSG)



DAOS-SEIS API



Benchmarking

Benchmarking is done by comparing a serial DSG API against the traditional seismic unix.

Data Description: All data were generated synthetically using seismic unix

Size	Trace Number	Samples Per Trace	FLDR	CDP
100 GB	12,999,000	2001	4,333 unique value, 3,000 trace per value	18,827 unique value, ~690 trace per value
1 TB	129,990,000	2001	43,330 unique value, 3,000 trace per value	174,815 unique value, ~740 trace per value
10 TB	1,331,127,000	2001	443,709 unique value, 3,000 trace per value	1,776,331 unique value ~750 trace per value

Benchmarking: Lenox cluster	DAOS Environment	GPFS Environment
Storage Nodes	4	4
	Cascade Lake-EP 6238 Gold (2 sockets)	Cascade Lake-EP 6238 Gold (2 sockets)
	8x P4610 1.6TB NVMe disks	8x P4610 1.6TB NVMe disks
	12x optane Pmem 128 GB	-
	2x EDR InfiniBand	2x EDR Infiniband
Computation Nodes	1	1
	Xeon Platinum 9242 (2 sockets)	Xeon Platinum 9242 (2 sockets)
	384 GB memory	384 GB memory
	1x HDR100 InfiniBand	1x HDR100 InfiniBand

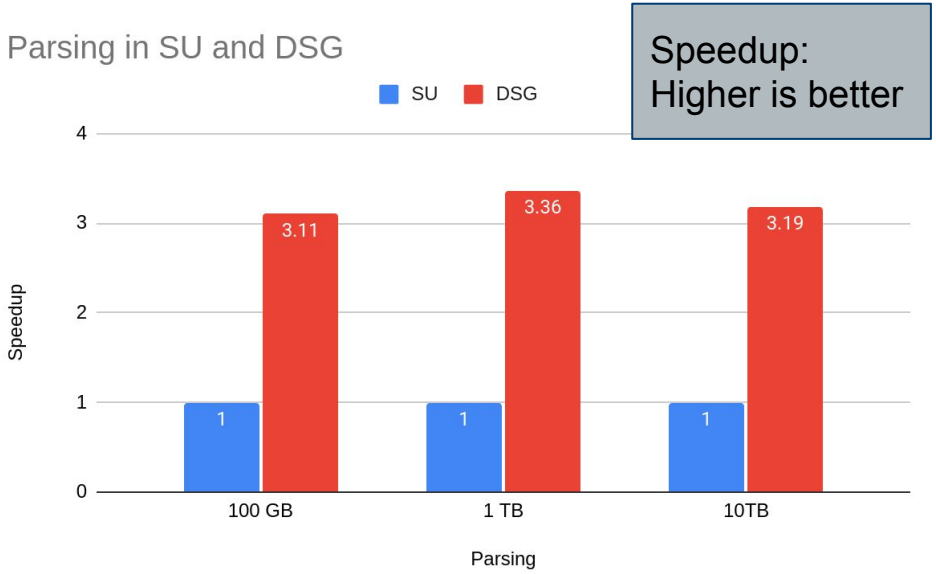
Benchmarking : Operations

Three main operations will be explored:

- **Parsing:**
  - Initial process executed before any seismic operation responsible for converting the segy file to a specific format.
    - SU : The segy file is parsed, processed then stored in the SU format.
    - DSG : The segy file is parsed, indexing of the traces is built, then data is stored in the DSG format.
- **Reading Shots Filtered By Metadata:**
  - Reading a range of the seismic data given a specific header range that is used as a selection criteria while fetching.
    - SU : SU read functionality passes over all the file, and only the requested range of data is returned.
    - DSG : Only the requested range of data is directly fetched and returned.
- **Sorting & Build indexing:**
  - Reading the seismic data in a specific sorting.
    - SU : All the su file is parsed, sorted using headers, then written back in su format.
    - DSG : Only headers are fetched, and indexing is built based on the sorting keys given.

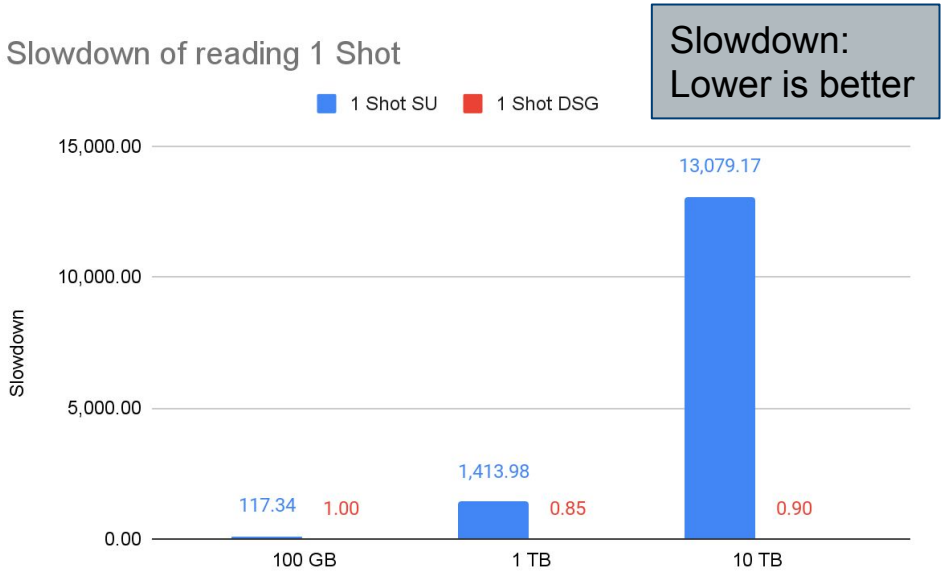
Benchmarking - Parsing

- It is noticed that parsing and transforming the segy file to the DSG format in comparison with transforming it to SU format is almost ~3x faster.
    - This is mainly due to the way DSG fetches and processes the traces in batches, while in SU each trace is fetched and processed individually.



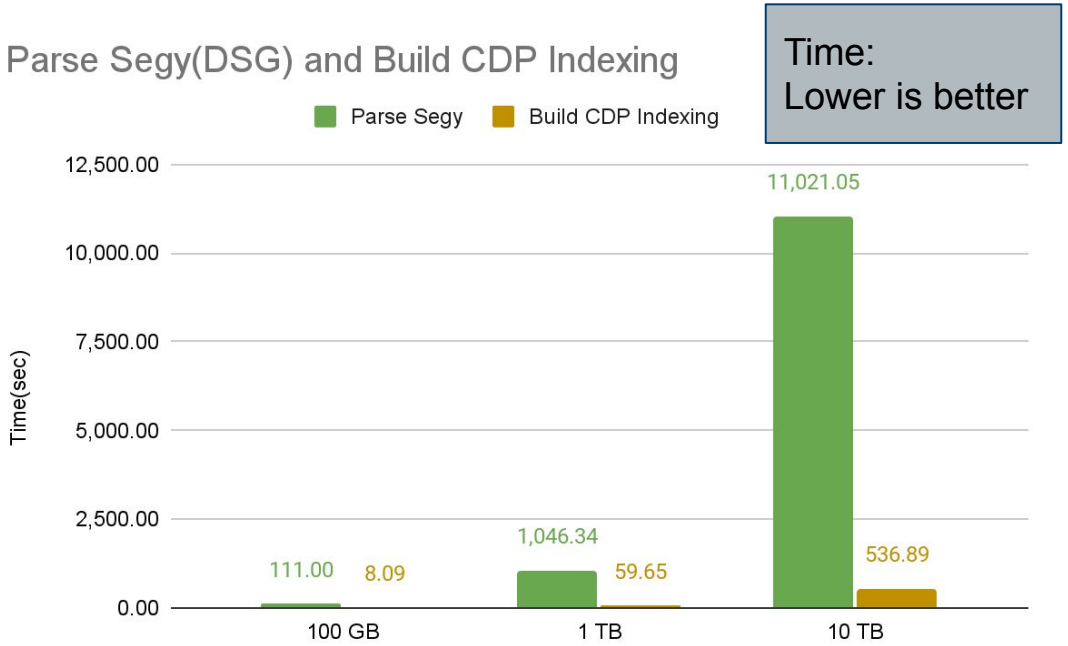
Benchmarking - Reading Shots Filtered By Metadata (FLDR)

- It is noticed that even with the increase in data size, the DSG doesn't incur any performance cost, and the performance is consistent.
- With seismic unix, due to the way it searches to read a specific shot, about 10x slowdown is noticed as data size increased 10x.



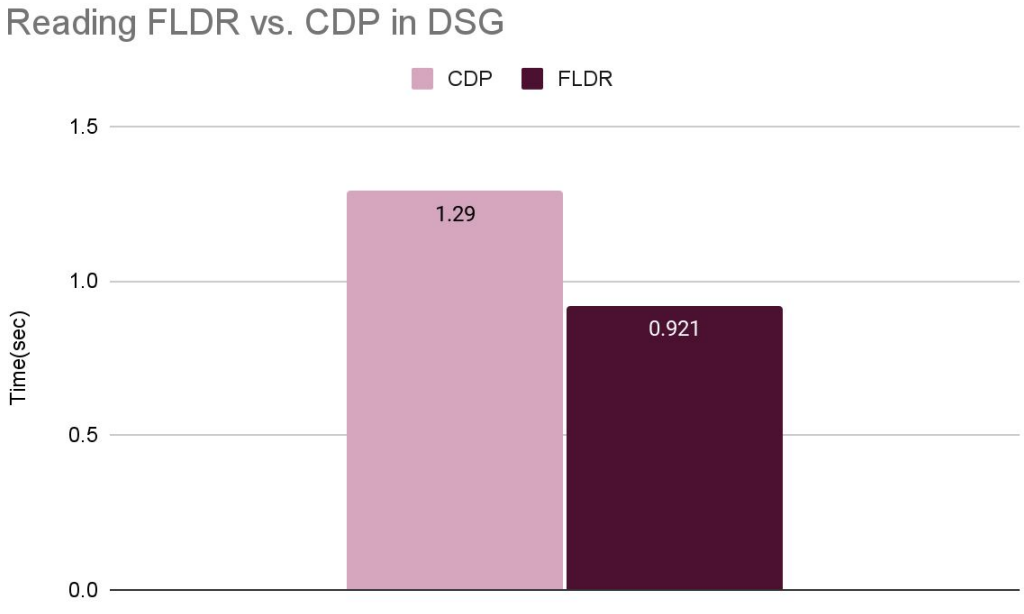
Benchmarking - Build Indexing

- Comparing the time for building indexes and parsing in DSG shows us that, it is always consistent, around 5% of the parsing time, providing a much faster way to access the same data in different orders.



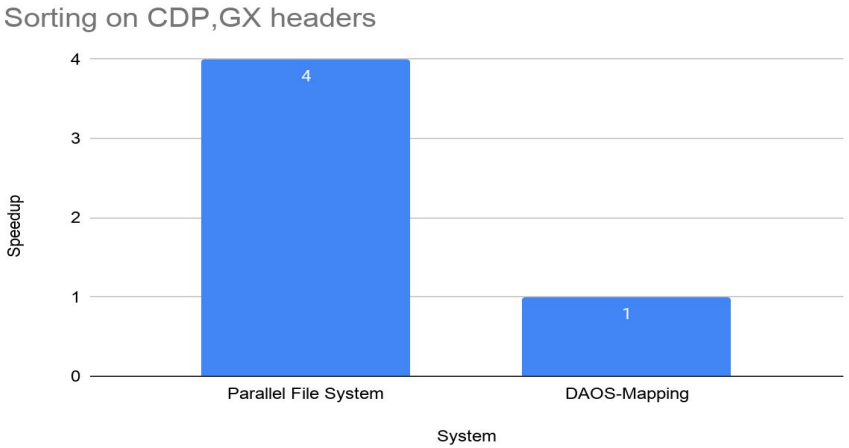
Benchmarking - Reading Shots Filtered By Metadata

- Building different indexes to the same dataset made the time of retrieval of one shot almost consistent in DSG.



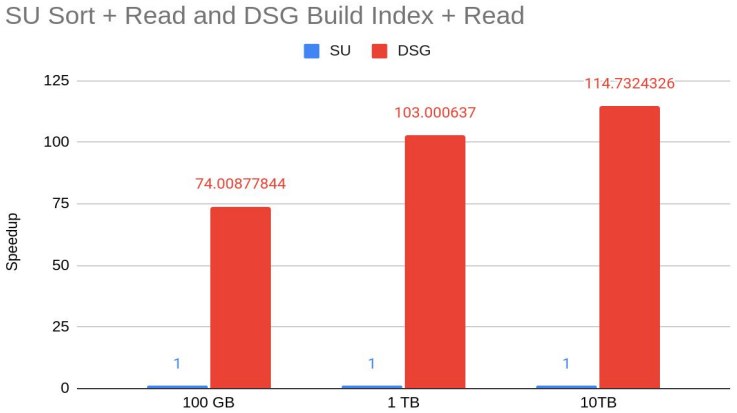
Benchmarking - Compared to Our Previous Release

Last Year's Results



Last year's results were held back by the fact that each of the traces' headers were read entirely and sequentially

This Year's Results



This year's results have improved greatly by completely replacing Seismic Unix's sort functionality through our Build Indexing functionality.

Conclusion & next steps

Utilizing the DSG API along with the DAOS storage model solves one of the major bottlenecks in the oil and gas industry, but also a room for optimization exists:

- Improve existing bottlenecks and further improve the DSG API.
- Explore the effect of parallelizing the DSG API.
- Integrate the DSG API with seismic processing applications.
- Compare the DSG performance with more optimized seismic IO libraries.

Resources

- Link to the DAOS-SEIS mapping wiki page:
  - https://daosio.atlassian.net/wiki/spaces/DC/pages/4853268687/DAOS-SEGY+Mapping
- Link to the open-source github repository of the DAOS seismic mapping:
  - https://github.com/daos-stack/segy-daos

THANKS