ExSeisFlow: An Extreme-Scale I/O Optimized Seismic Workflow Library

Meghan Fisher[a,b], Pádraig Ó Conbhuí[a,b], Cathal Ó Brion[a,b], Seán Delaney[c], Gareth O'Brien[c], Jean-Thomas Acquaviva[d], James Coomer[d], Steven Dagg[c], Ruairi Short[a]

a. The Irish Centre for High-End Computing (ICHEC), b. Lero - The Irish Software Engineering Research Centre, c. Tullow Oil Limited, d. DDN Storage

Large file size exacerbates I/O and memory bottlenecks

The increasing size of seismic files (100's of GB to 10's of TB) means non-collective I/O for parallel workflows is becoming less efficient, or perhaps nonviable, due to memory limits as a result of hardware.

Different data access patterns (both contiguous and non-contiguous) in common file formats, like SEG-Y, increases development complexity and time, as well as overall runtime.

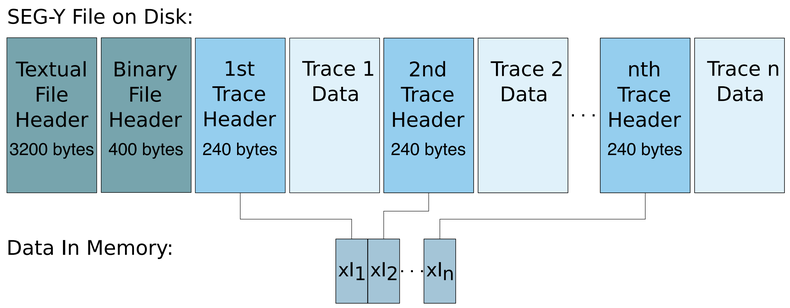


Figure 1: Data access pattern for a header values for all traces (e.g. crossline position) in a SEG-Y file. While single trace access in this file format is contiguous, accessing multiple traces or header values for multiple traces is non-contiguous, which is less efficient than sequential access.

ExSeisFlow is a user-focused, easy to use parallel library

ExSeisFlow is a parallel workflow library designed to target pre-stack pre-processing with special focus on large datasets. It is an operations driven library that reads and writes standard and custom SEG-Y files, with support of both proprietary and other open source (e.g. HDF5) file formats in development. It has C and C++ implementations with python available in Q2 2018. It is licensed under LGPLv3.

Operations are performed on single traces, individual gathers, and across entire surveys, including sorting, binning, filtering, and transforming. These handle all distributed communication internally, meaning developers do not need to call any MPI routines.

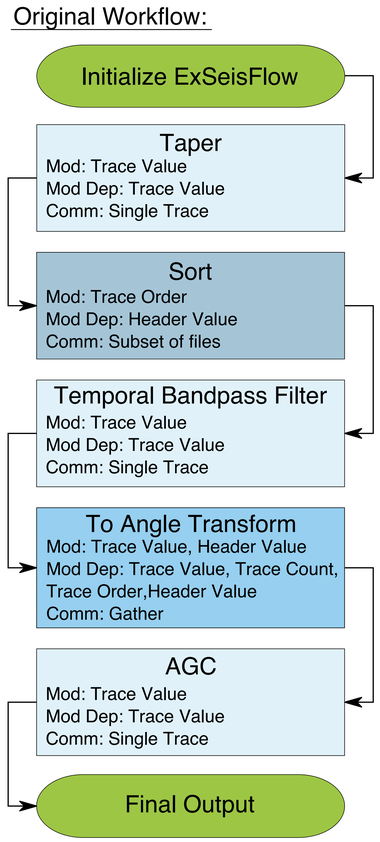
ExSeisFlow is available as part of the ExSeisDat project and can be downloaded at http://github.com/ICHEC/ExSeisDat

```
auto piol = ExSeis::New()
Set set(piol, "In*.segy", "Out.segy")
set.sort(PIOL_SORTTYPE_OffLine)
return 0;
```

Download Link:

Figure 2: Example of ExSeisFlow C++ workflow code that concatenates files and sorts them. All I/O is performed when the destructor is called rather than using explicit read and write commands. Similarly, all workflow operations are not performed when the function is called but rather directly before output. ExSeisFlow uses commands intuitive to geophysicists and function calls designed to further simplify library use.

Memory and I/O are implicitly optimized via an internal queuing system



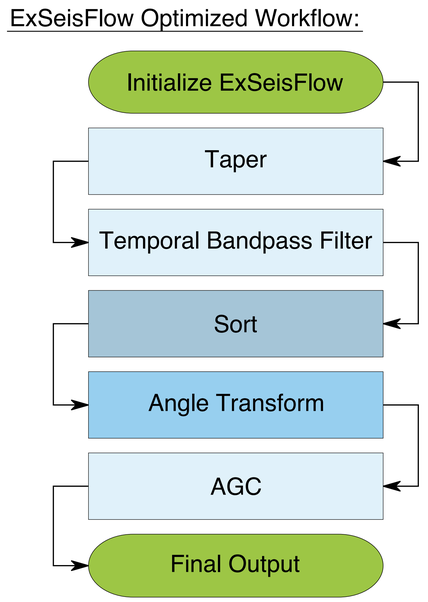


Figure 3: Example ExSeisFlow workflow that applies a taper to each trace, sorts them into inline/crossline order, applies a temporal bandpass filter, transforms the data set to the angle domain, and finally applies automatic gain control for visualization. Based on the modifications and modification dependencies of each operation, ExSeisFlow is able to reorganize the workflow to optimize for memory and I/O.

ExSeisFlow leverages a parallel I/O library (ExSeisPIOL) to implicitly handle all reading and writing required by any operation, internally determining what data is needed for the operation, the access pattern for that data, and the amount of memory needed. ExSeisFlow has an internal queuing system to optimize I/O and memory within the workflow. When an operation is called within a workflow code, it is queued and performed when the destructor is called. Each operation has a series of tags that identify the data required for the operation, what is modified, and what type of communication is required. The internal queuing system allows for smart I/O and memory caching by:

- Storing header and trace data when used again later in the workflow and outputting/discarding data when no longer used
- Optimizing workflow order so that operations that require the same data can be performed sequentially, saving memory and superfluous MPI communications
- Maintaining operation order when there is a data dependency

Modification	Modification Dependency	Communication
Add Trace Delete Trace Change Trace Value Reorder Traces Change Header Value	Trace Value Header Value Trace Order Trace Count	Single Trace Gather Subset of Files Entire Data Set

Table 1: The three different categories of operation tags with lists of possible tags for each category. An operation can have multiple modification and modification dependency tags, but can only one communication tag. Operations are reordered based on their communication type, unless there is a modification that affects a future operation dependency.

ExSeisFlow and burst buffer technology drastically improves I/O perfomance

Leveraging next generation burst buffer technology can further improve the I/O performance of ExSeisFlow. Using DDN's Infinite Memory Engine (IME) requires no change in compiling or using the library. The library has been benchmarked using a simple sorting workflow on files up to 20 TB.

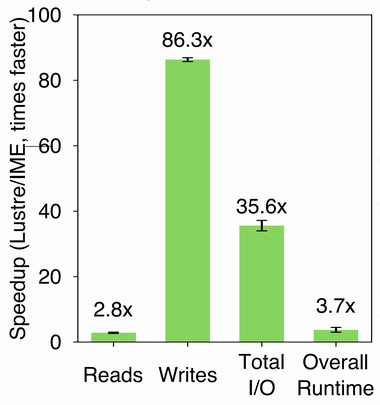


Figure 4: Average speedup (Lustre/IME) for sorting a file ranging from 100 GB to 20 TB in size. The 2.8x speedup in read time is indicative of the SSD cache of IME. The 86.3x speedup in write time can be attributed how IME handles non-contiguous writes, which are needed when writing SEG-Y files to disk.

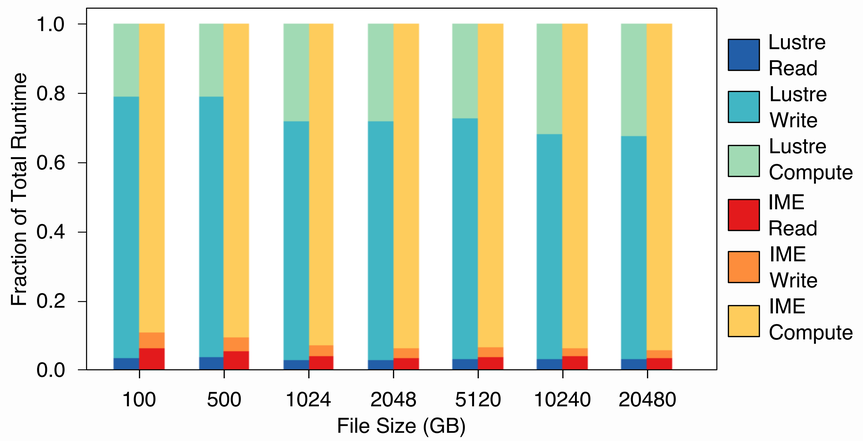


Figure 5: Ratio of read, write, and compute time to overall runtime on a Lustre parallel file system and a Lustre+IME parallel file system. Note that the compute time remains the same between Lustre and IME runs, while the compute percentage increases. By leveraging IME, the sorting workflow goes from an I/O bound problem to a compute bound one.

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie). ICHEC is affilliated with the National University of Ireland Galway (NUIG)