A Best Practice Analysis of HDF5 and NetCDF-4 Using Lustre

Christopher Bartz[1], Konstantinos Chasapis[2], Michael Kuhn[2], Petra Nerge[2], Thomas Ludwig[1]

[1]Deutsches Klimarechenzentrum

[2]University of Hamburg

2015-07-15

Agenda

1. Motivation
2. Background
3. Evaluation
4. Conclusion

Agenda

1 Motivation

2 Background

3 Evaluation

4 Conclusion

Parallel I/O performance

- Scientific applications store data in various formats
- HDF5 and NetCDF-4 are widely used data formats, surrounded by high-level I/O interfaces
- I/O performance can be crucial to overall performance
- I/O can be performed in parallel
- Suboptimal I/O performance depending on the application's access patterns

Goal of analysis

- Enable high performance I/O using HDF5 and NetCDF-4
- Provide best practices for using I/O
- Discover deficiencies and provide enhancements
- Therefore, analysis of different access patterns and I/O configurations

Agenda

1 Motivation

2 Background

3 Evaluation

4 Conclusion

Overview I

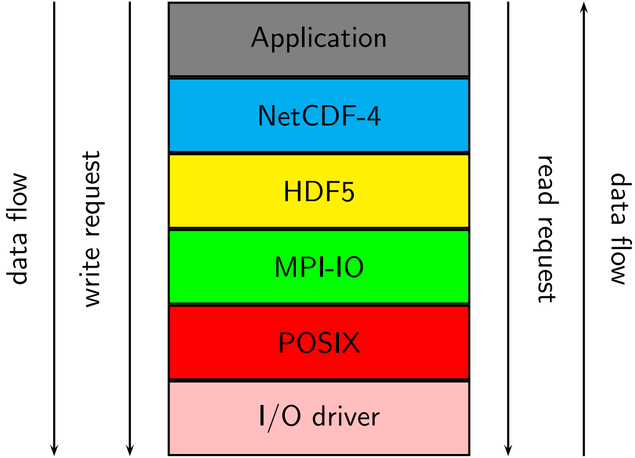


Figure: Involved I/O layers and data flow

Lustre

- Stores data in a distributed manner
- File is split up into multiple objects ("stripes")
- Stored on different Object Storage Targets (OSTs)
- Distribution of the stripes among the OSTs in a round-robin fashion
- Clients use standard POSIX I/O system calls

Lustre: File striping

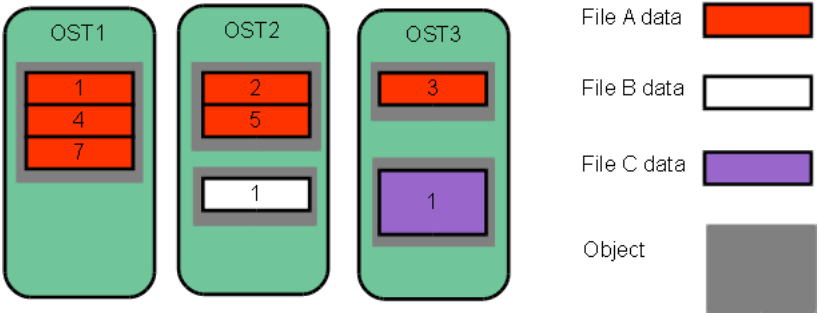


Figure: File striping[1]

[1] http://build.whamcloud.com/job/lustre-manual/lastSuccessfulBuild/artifact/lustre_manual.xhtml

HDF5

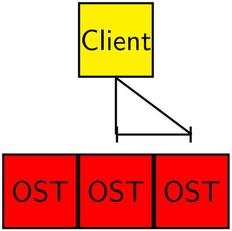
- Stores data in multi-dimensional arrays
- Dimensions can also be unlimited
- Data can be stored contiguously in one large block in the file
- Data can also be stored using chunked layout:
  - Data is split into multiple pieces
  - Written into independent locations in the file
  - Locations are stored in a B-tree in the header of the data
  - Required for advanced features like compression

HDF5

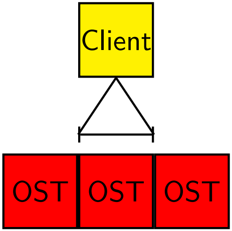
- Stores data in multi-dimensional arrays
- Dimensions can also be unlimited
- Data can be stored contiguously in one large block in the file
- Data can also be stored using chunked layout:
  - Data is split into multiple pieces
  - Written into independent locations in the file
  - Locations are stored in a B-tree in the header of the data
  - Required for advanced features like compression

Alignment

- HDF5 provides a routine that aligns the address of the file objects to particular boundaries
- Lustre stripes are useful boundaries



Unaligned



Aligned to Lustre stripes

NetCDF-4

- Like HDF5, stores data in multi-dimensional arrays
- Used in the scientific community, especially in climatology, meteorology and oceanography
- NetCDF-4 directly uses HDF5; NetCDF-4 files are HDF5 files
- NetCDF-4 does not provide a routine to align the file objects

Agenda

1 Motivation

2 Background

3 Evaluation

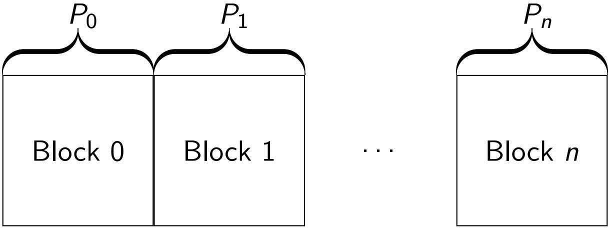
4 Conclusion

Experimental design

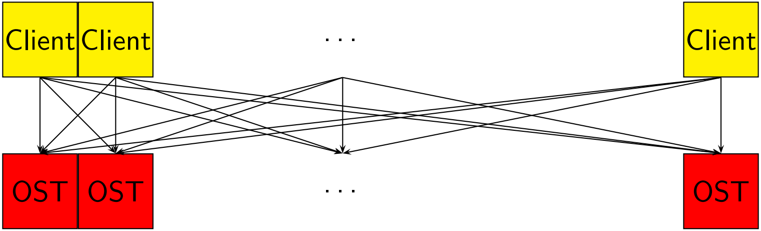
- 10 OSTs, each with single hard drive
- 10 client nodes
- Interconnected via Gigabit Ethernet, maximum performance: $\approx 1,125\text{MiB/s}$
- 3 repetitions write / read per I/O configuration, plots show mean
- Write/read 20 GiB per node (exceeds available memory)
- Accesses are aligned to the Lustre stripe boundaries, for NetCDF-4 we are using the original and an alignment-enabled version

Disjoint pattern: Overview

Each client accesses a large contiguous region.

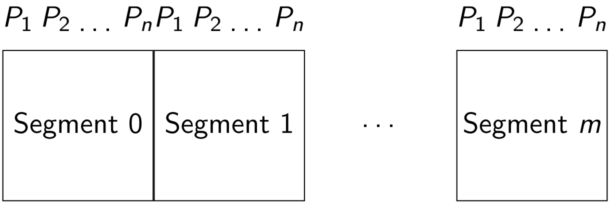


This is called all-to-all pattern:



Interleaved pattern: Overview

Each client accesses a non-contiguous region.



P_1 P_2 . . . P_n P_1 P_2 . . . P_n P_1 P_2 . . . P_n

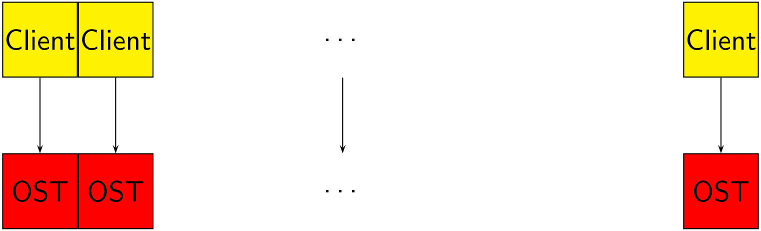
Segment 0

Segment 1

· · ·

Segment m

1-OST pattern:



Client

Client

· · ·

Client

OST

OST

· · ·

OST

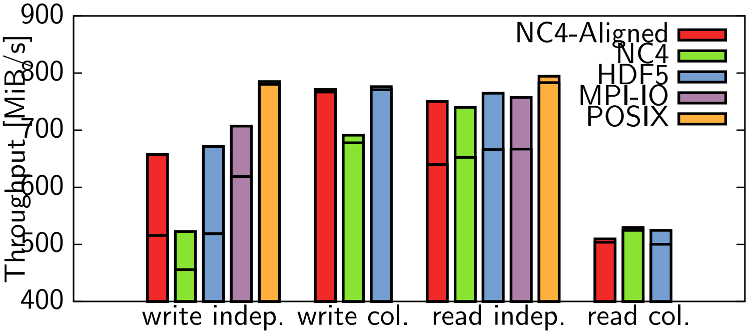


Figure: Disjoint pattern

- Maximum and minimum values shown
- Lower layers yield higher performance
- Overhead induced by libraries reduces performance

Discussion

- Contention on OSTs and network resources
- Results much lower than the practical maximum
- High variation when using independent I/O, due to lack of synchronisation

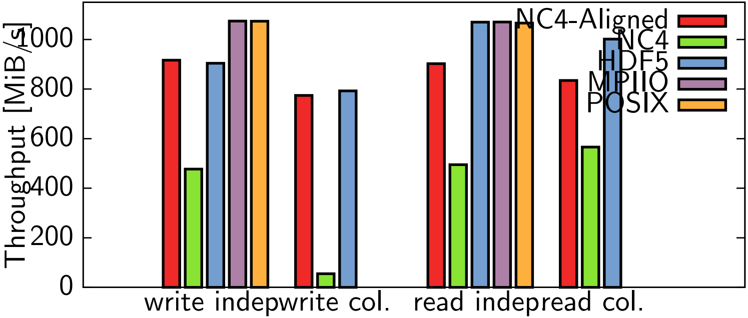


Figure: 1-OST pattern

- Performance much more stable than with disjoint pattern

Discussion

- 1-OST pattern with POSIX or MPI-IO almost the practical maximum
- HDF5 similar when reading
- HDF5 write independent better than the disjoint pattern
- NetCDF-4 API without alignment patch much worse than the other APIs, because of unaligned access

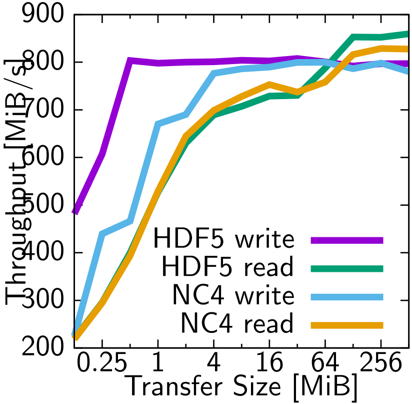


Figure: Varying transfer size

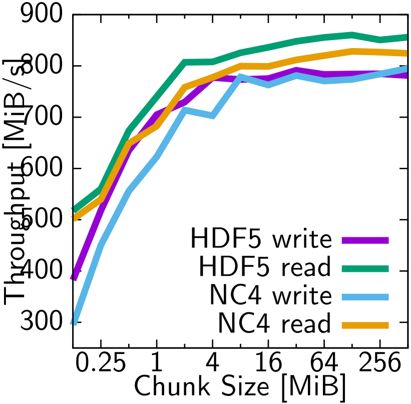


Figure: Chunked layout

- HDF5 scales better with the transfer size

Discussion

- The highest throughput is achieved with large transfer sizes
- Chunked I/O benefits from large chunk sizes
- Required sizes often much larger than practically useful

Agenda

1 Motivation

2 Background

3 Evaluation

4 Conclusion

Best practices: HDF5

If chunking is not required, use contiguous layout with

- 1-OST pattern with I/O accesses aligned to the Lustre stripes
- Independent I/O

If chunking is required (for example, due to compression),

- Disjoint pattern
- Collective I/O
- Large chunk size (relative to file size)
- Large transfer size (relative to stripe size and amount of OSTs)

Best practices: NetCDF-4

If chunking is not required, use contiguous layout with

- Disjoint pattern
- Collective I/O
- Large transfer size

If chunking is required (for example, due to compression),

- Disjoint pattern
- Collective I/O
- Large chunk size
- Large transfer size

Summary

- Disjoint pattern: Figures significantly lower than practical maximum performance
- Interleaved pattern: 1-OST pattern achieves maximum performance with POSIX and MPI-IO
- Performance benefits from large transfer and chunk sizes
- I/O performance very sensitive to correct access pattern
- Manual tuning by application developers necessary

NetCDF-4 enhancements

- Implemented alignment for NetCDF-4: reevaluation showed improved figures
- We have opened a bug report for NetCDF-4
- As far as we know, functionality still not available
- HDF5 requires setting explicit alignment by the developer
- Should probably be enhanced to automatically figure out alignment based on underlying file system

ISC 2015 Student Cluster Competition

Visit us at booth 418 and vote for us! ☺
Twitter: @UHH_ISC_SCC