An I/O Study of ECP Applications

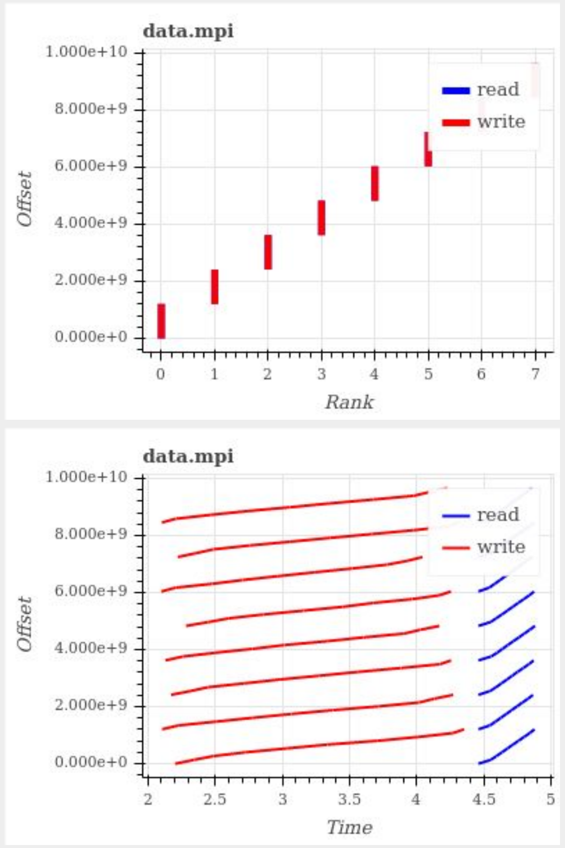
Setup

8 Processes: 4 nodes x 2 ranks/node

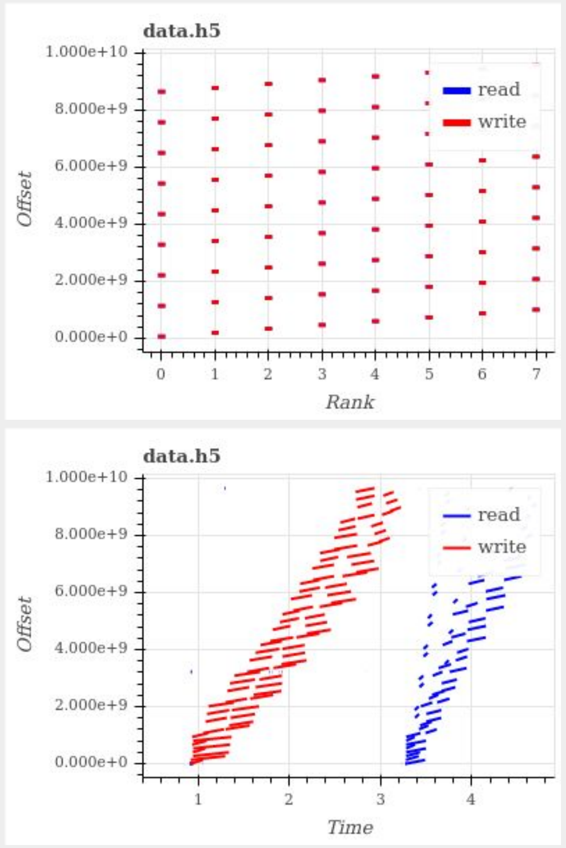
1GB per variable, 9GB in total

1. Individual dataset (9 dataset, one for each variable)
2. Compound data type
3. With new HDF5 API - AllocaMulti

HACC-IO: MPI vs HDF5, why HDF5 is slow

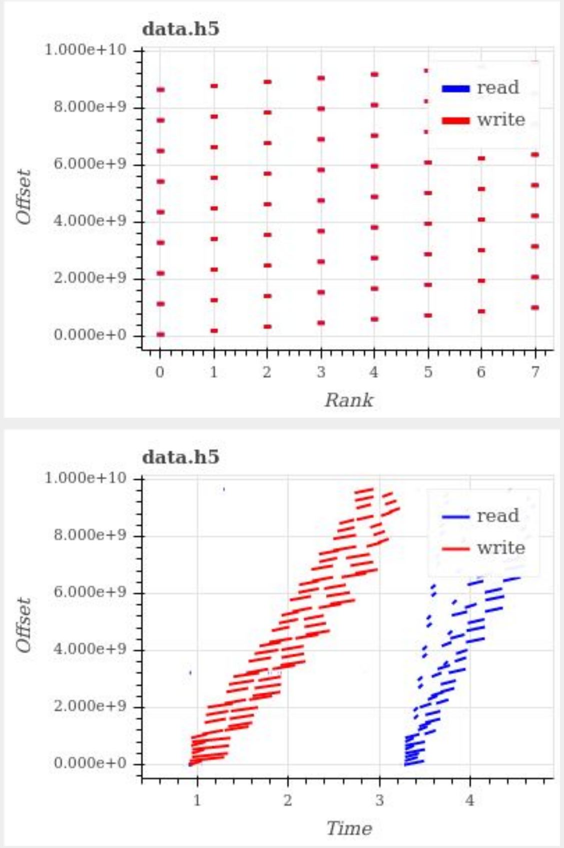


MPI

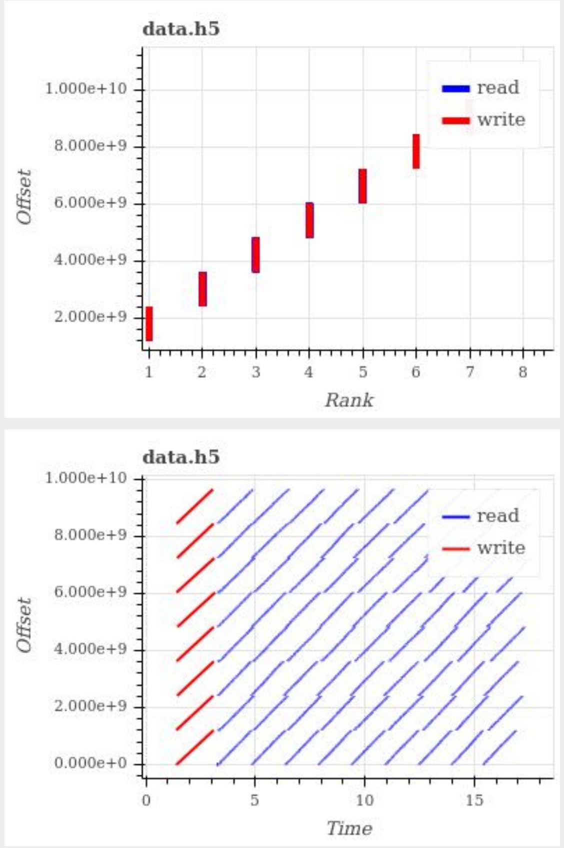


HDF5 with individual dataset

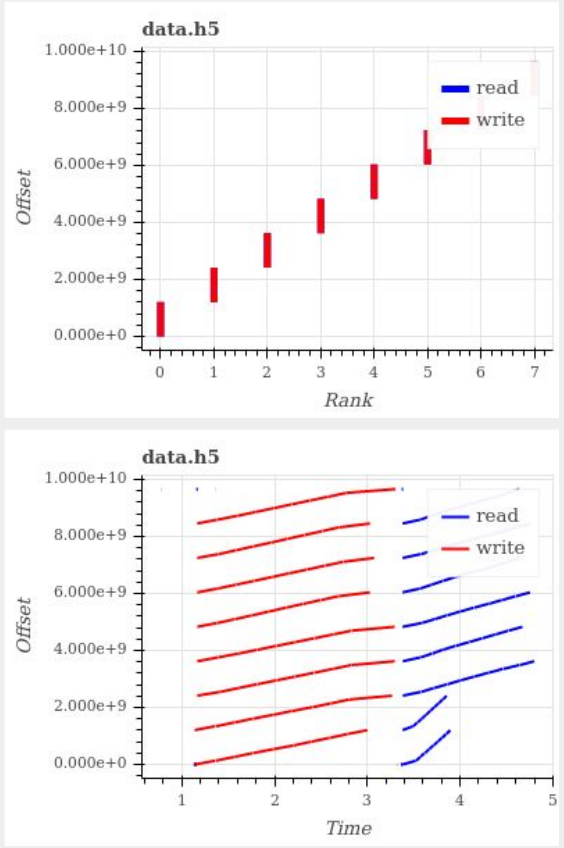
HACC-IO: HDF5 using different I/O methods



Individual Dataset



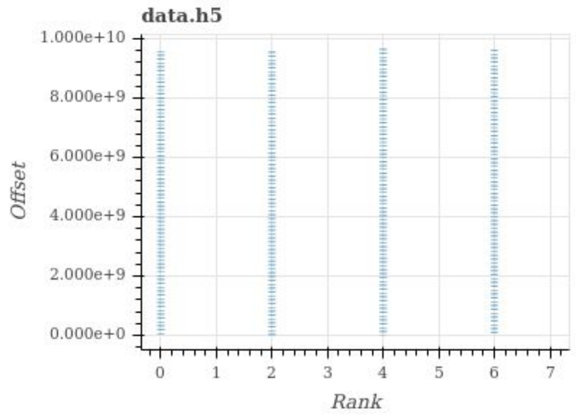
Compound Datatype



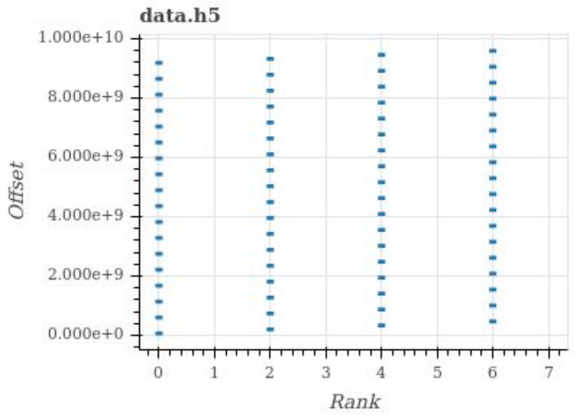
Alloca Multi

HACC-IO: will collective I/O help with individual dataset?

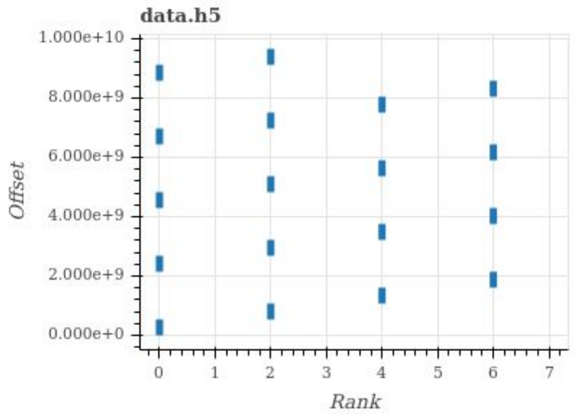
HDF5 with individual dataset but with collective I/O enabled



Stripe Size: 1M,
Stripe Count: 4



Stripe Size 128M
Stripe Count: 4

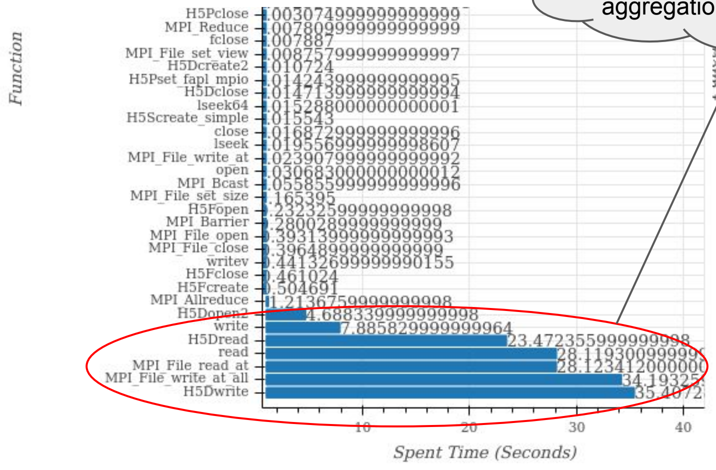


Stripe Size 512M
Stripe Count: 4

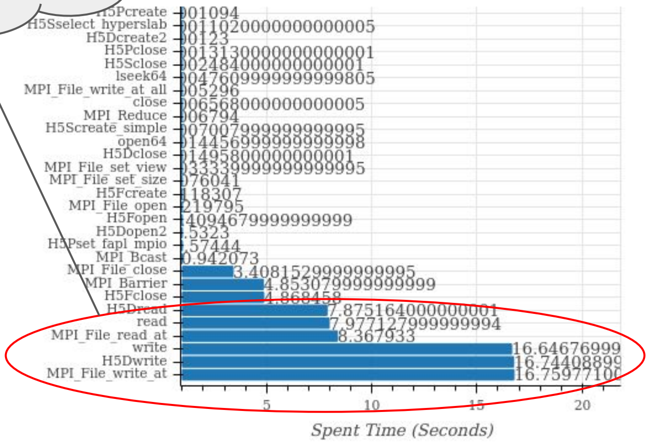
HACC-IO: Is collective I/O always better?

- When the request size is big, the collective communication overhead increases and the benefits from collective I/O becomes limited.

The collective write is indeed faster. But the overhead of aggregation?



Collective write
Stripe size:128M



Independent write

Recorder

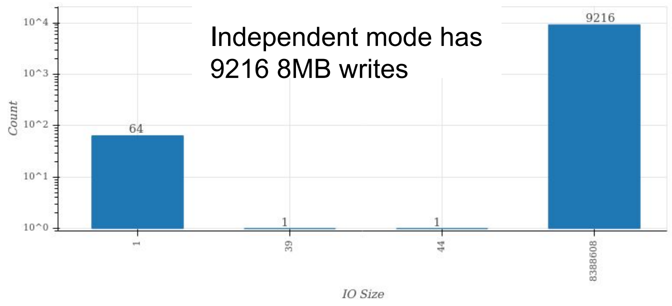
- Multi-level I/O tracing library that captures function calls from HDF5, MPI and POSIX.
- No aggregation, it keeps every function and its parameters. Useful to exam access patterns.
- Built-in visualizations for access patterns, function counters, I/O sizes, etc.
- Also reports I/O conflicts such as write-after-write, write-after-read, etc. Useful for consistency semantics check (File systems with weaker consistency semantics).

Wang, Chen, Jinghan Sun, Marc Snir, Kathryn Mohror, and Elsa Gonsiorowski. "Recorder 2.0: Efficient Parallel I/O Tracing and Analysis." In IEEE International Workshop on High-Performance Storage (HPS), 2020.
https://github.com/uiuc-hpc/Recorder

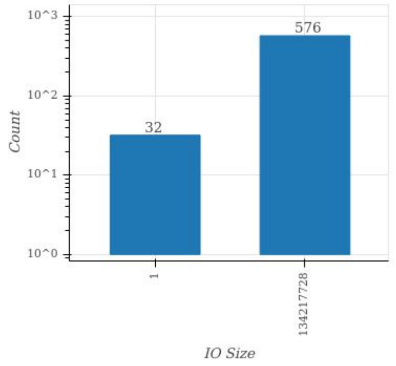
Please ignore the rest slides.

HACC-IO: Is collective I/O always better?

- Strip size of 128M

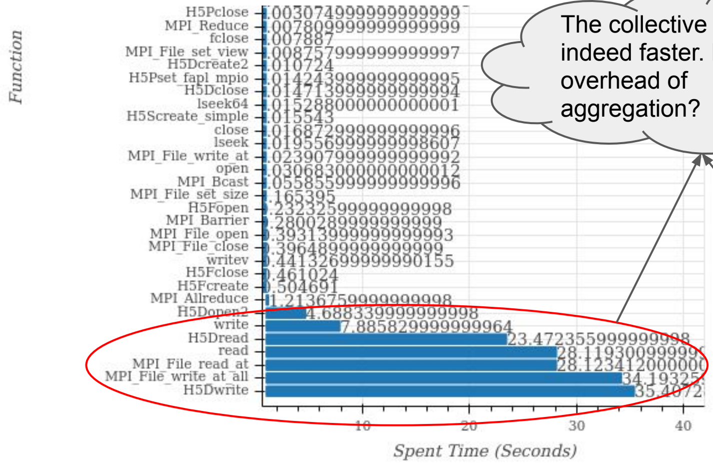


Independent mode has 9216 8MB writes



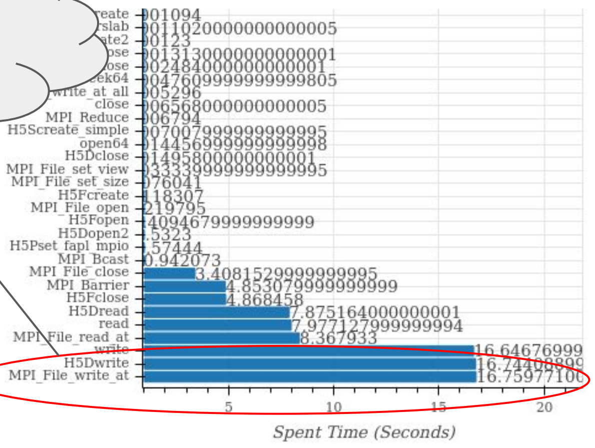
Collective mode has 576 128M writes

Collective



Collective

The collective write is indeed faster. But the overhead of aggregation?



Setup

1024 Processes: 32 nodes x 32 ranks/node

HACC-I/O: write only. HDF5 collective metadata is **disabled**.

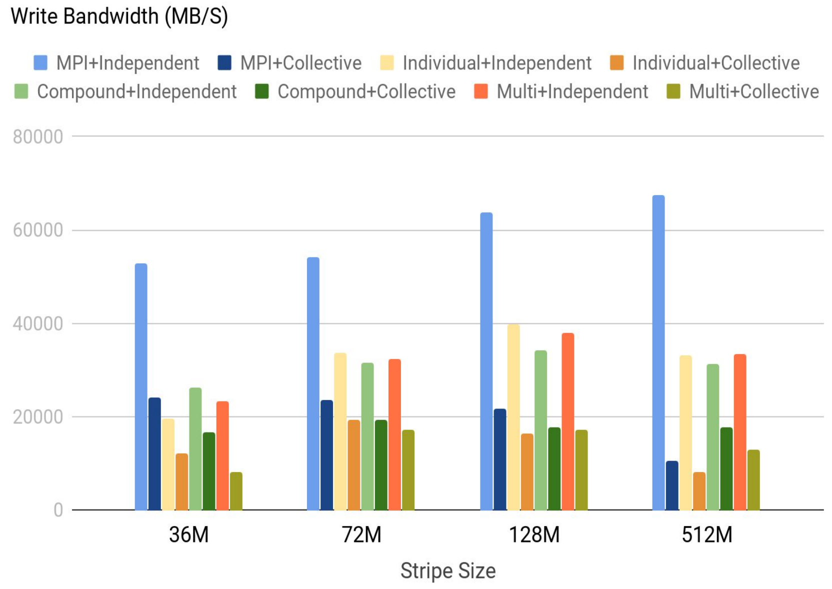
1. Compound data type
2. Individual dataset (9 dataset, one for each variable)
3. With new HDF5 API Multi Allocation

Each of the above three methods are run with both independent and collective MPI-IO mode. So total of 6 runs.

Aggregated write bandwidth with different stripe size

1024 Processes: 32x32

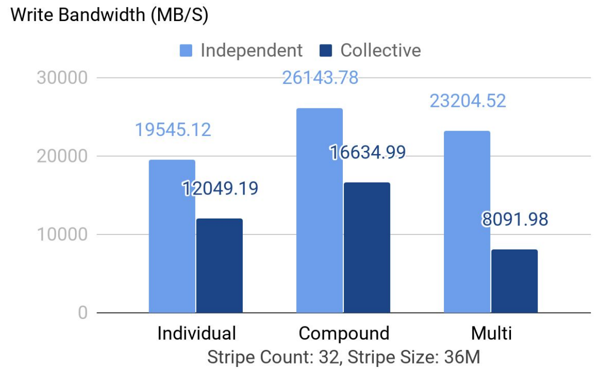
8GB per variable, 72GB total.

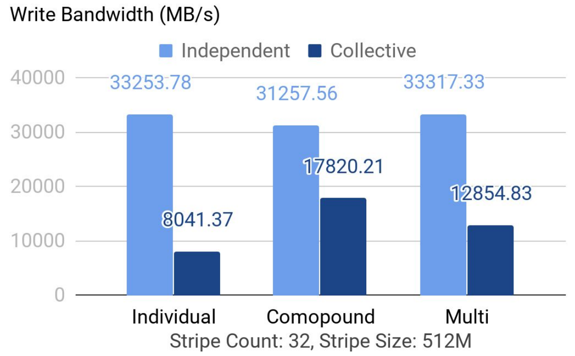


Aggregated write bandwidth with different stripe size

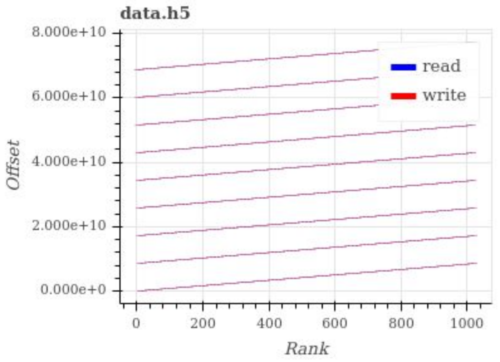
1024 Processes: 32x32

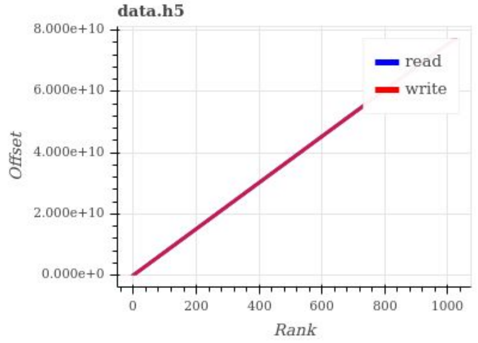
8GB per variable, 72GB total.

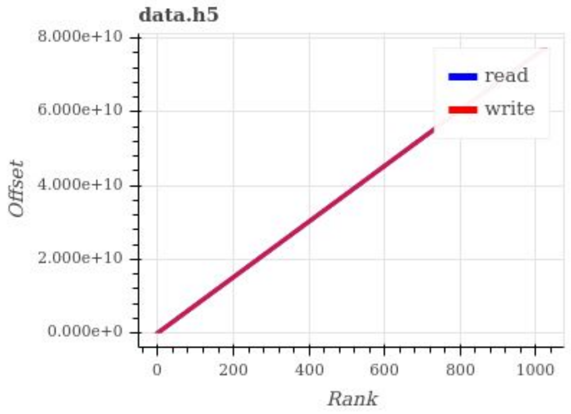


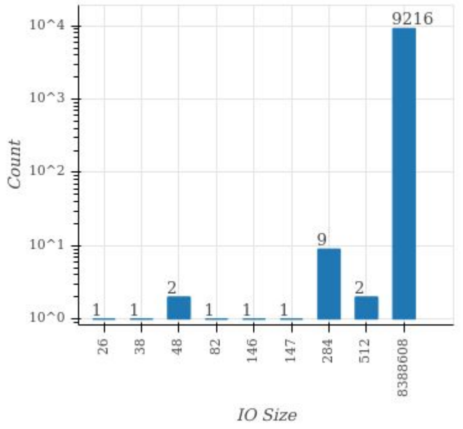


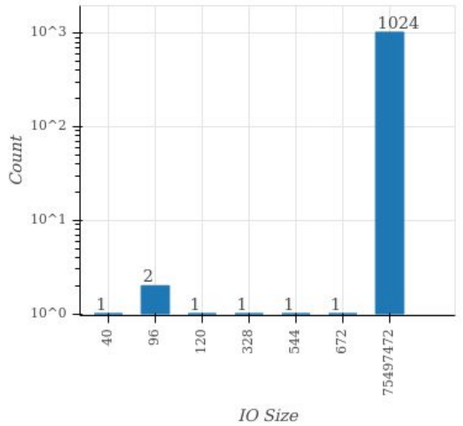
Independent

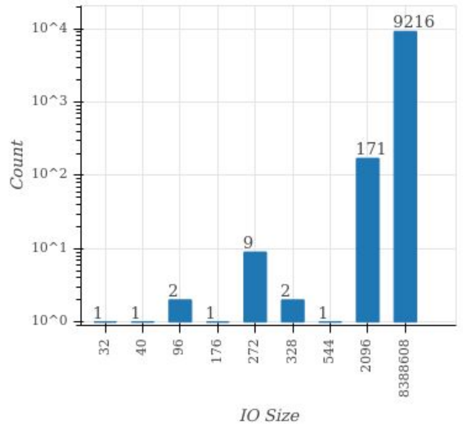




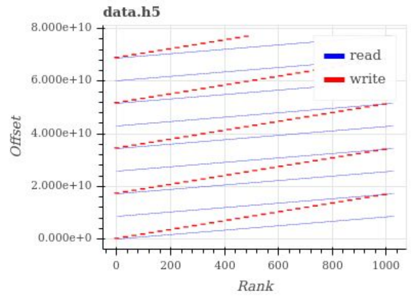


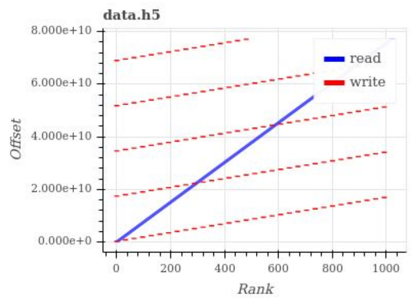


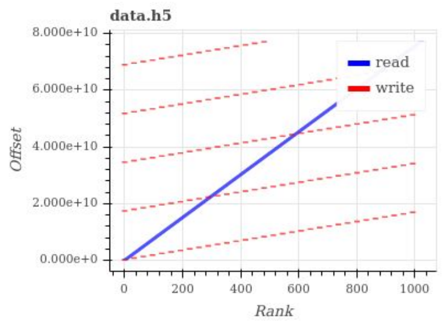


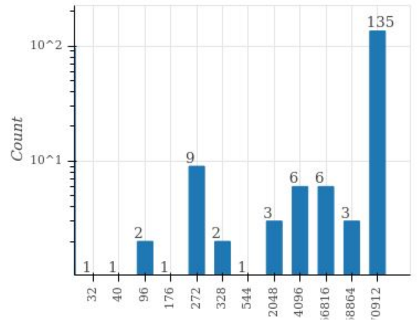


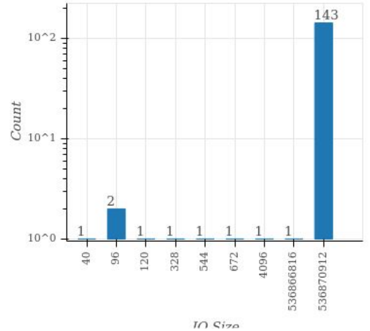
Collective

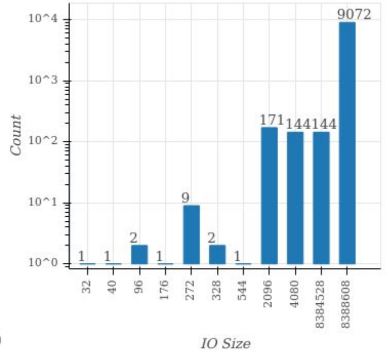

data.h5
read
write
Offset
Rank
8.000e+10
6.000e+10
4.000e+10
2.000e+10
0.000e+0
0
200
400
600
800
1000

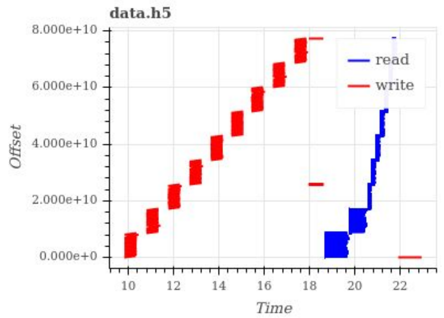

data.h5
read
write
Offset
Rank
8.000e+10
6.000e+10
4.000e+10
2.000e+10
0.000e+0
0
200
400
600
800
1000

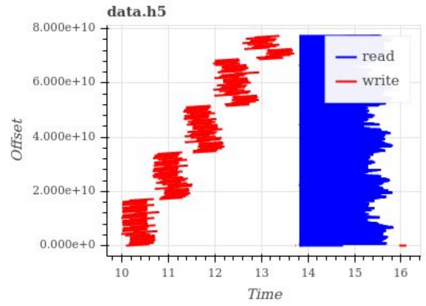

data.h5
read
write
Offset
Rank
8.000e+10
6.000e+10
4.000e+10
2.000e+10
0.000e+0
0
200
400
600
800
1000

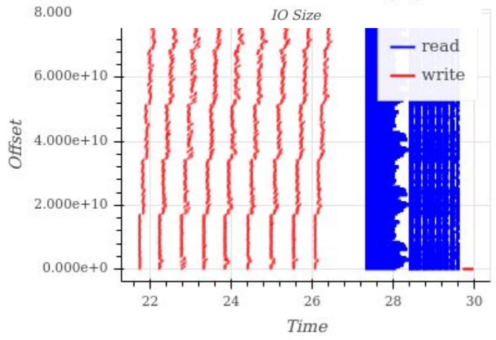

Count
10^2
10^1
1
1
2
1
9
2
1
3
6
6
3
135
32
40
96
176
272
328
544
2048
4096
6816
8864
70912

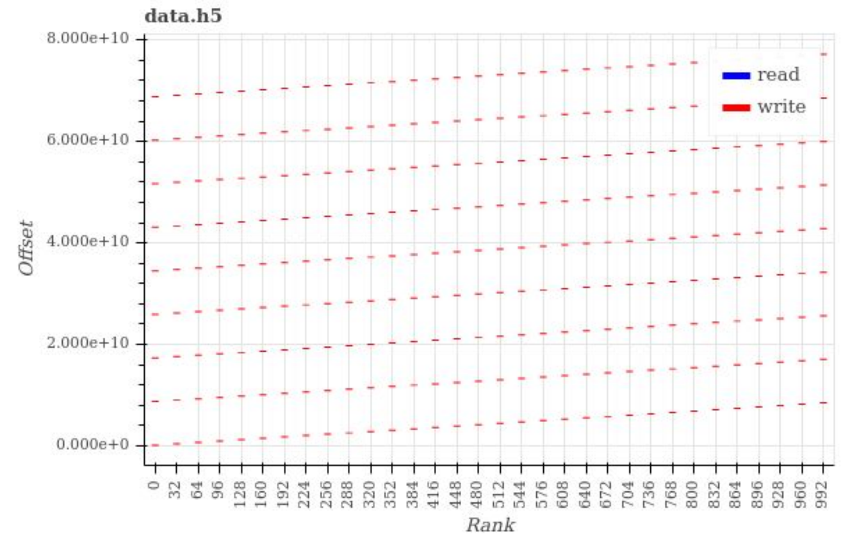

Count
10^2
10^1
10^0
1
2
1
1
1
1
1
1
143
40
96
120
328
544
672
4096
536866816
536870912
IO Size


Count
10^4
10^3
10^2
10^1
10^0
1
1
2
1
9
2
1
171
144
144
9072
32
40
96
176
272
328
544
2096
4080
8384528
8388608
IO Size


data.h5
read
write
Offset
Time
8.000e+10
6.000e+10
4.000e+10
2.000e+10
0.000e+0
10
12
14
16
18
20
22


data.h5
read
write
Offset
Time
8.000e+10
6.000e+10
4.000e+10
2.000e+10
0.000e+0
10
11
12
13
14
15
16


8.000
read
write
Offset
Time
6.000e+10
4.000e+10
2.000e+10
0.000e+0
22
24
26
28
30



HDF5 Individual Dataset with Collective MPI-IO