TOWARDS AUTOTUNING HDF5 WITH USER INTENT

Hariharan Devarajan, Gerd Heber, and Kathryn Mohror

hariharandev1@llnl.gov

2023 HDF5 User Group (HUG) Meeting

The HDF Group

Lawrence Livermore National Laboratory

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract LLNL-PRES-852846.

USER INTENT

User Intent is defined as **"why", "what",** and **"how"** users perform certain I/O operations

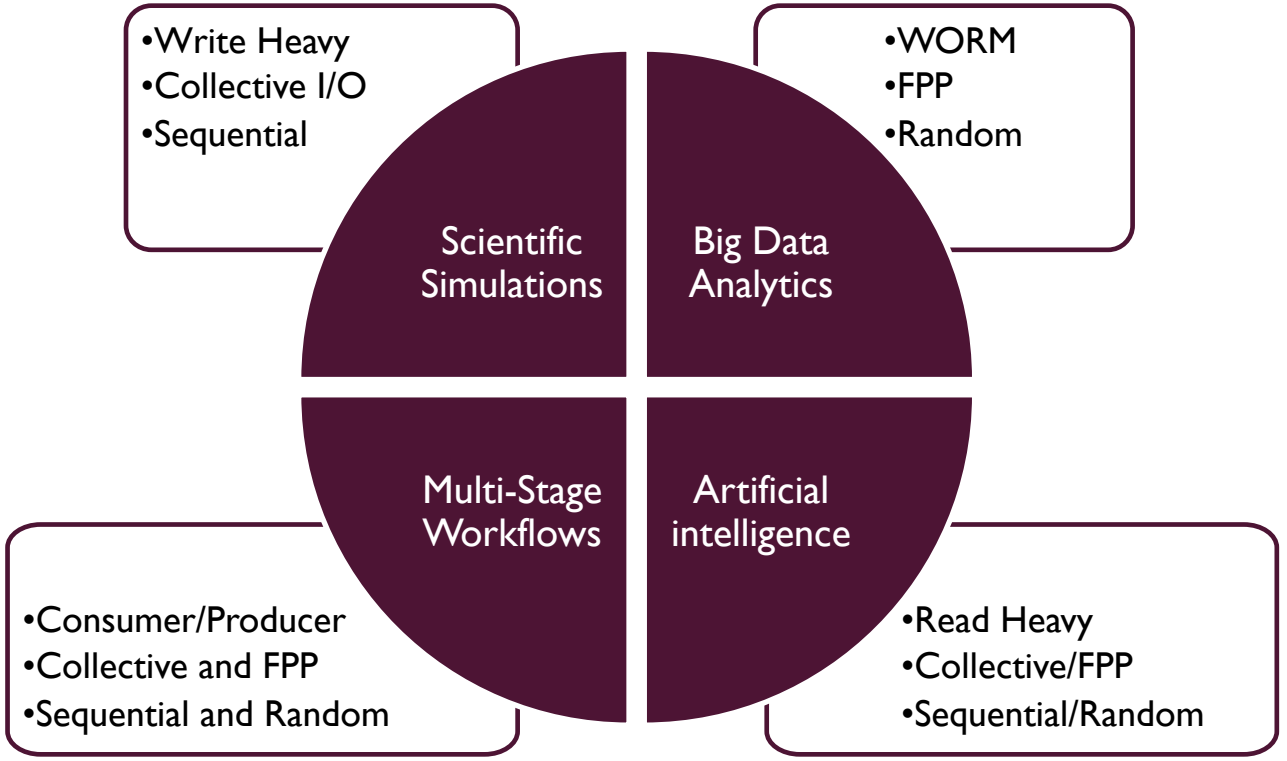
	What **IS** a User Intent		What **IS NOT** a User Intent	
What	**Why**	**How**	**Why**	**How**
HDF5 File	Checkpoint a process	Using FPP on PFS	Cache the file	Enable chunk cache
HDF5 Dataset	Read a sparse dataset	Using hyperslab in HDF5	Perform redundant I/O	Enable data sieving
POSIX File	Read samples from file system	32 samples per file in 1024 files with MPI-IO	Preload data	Use stage-in interface of bb.

We can think of Intents trickling down from the workload through the I/O software stack

DIVERSITY OF USER INTENT

- HPC workloads have moved beyond traditional simulations towards other workloads
  - This introduces a plethora of new I/O behaviors
  - These I/O behaviors need to be handled by storage systems and middleware libraries.

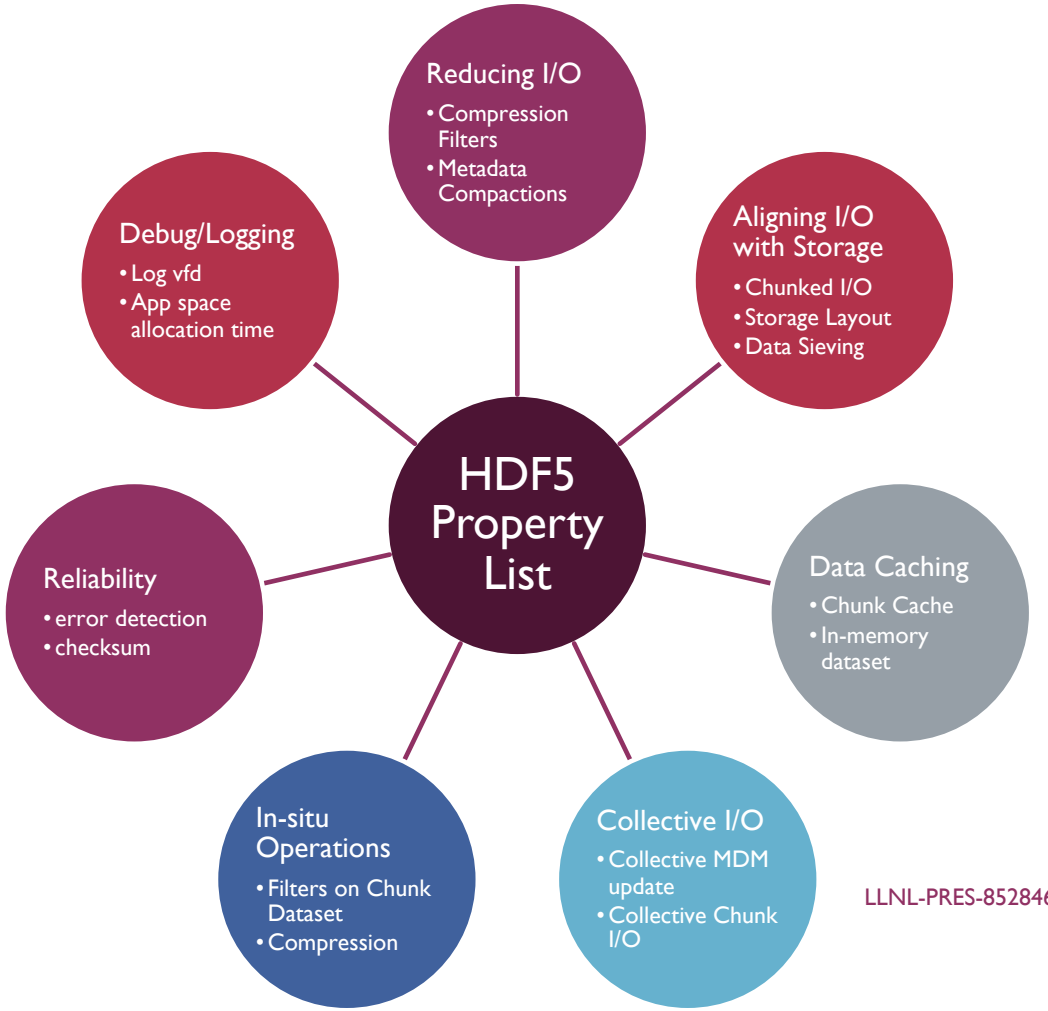
In response to changing user intents, I/O stacks try to become more adaptable through configurations



CONFIGURABILITY OF THE HDF5 LIBRARY

- HDF5 supports Property Lists to adapt to changing I/O requirements at runtime
- These property lists are used to enable various I/O features/optimizations within HDF5
  - There are over 100 properties users can set to optimize their workload.
    - For several reasons, that's a problem!

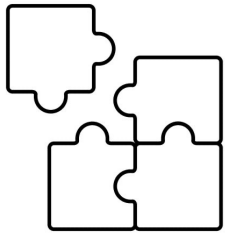
Through these property lists, HDF5 can support a wide variety of use-cases.



COMPLEXITY OF SETTING CONFIGURATIONS

User Intent

- FPP/Shared/Prod-Con
- Sequential/Random
- WO/RO/WORM/meta
- Small VS. Large I/O
- Reliability level

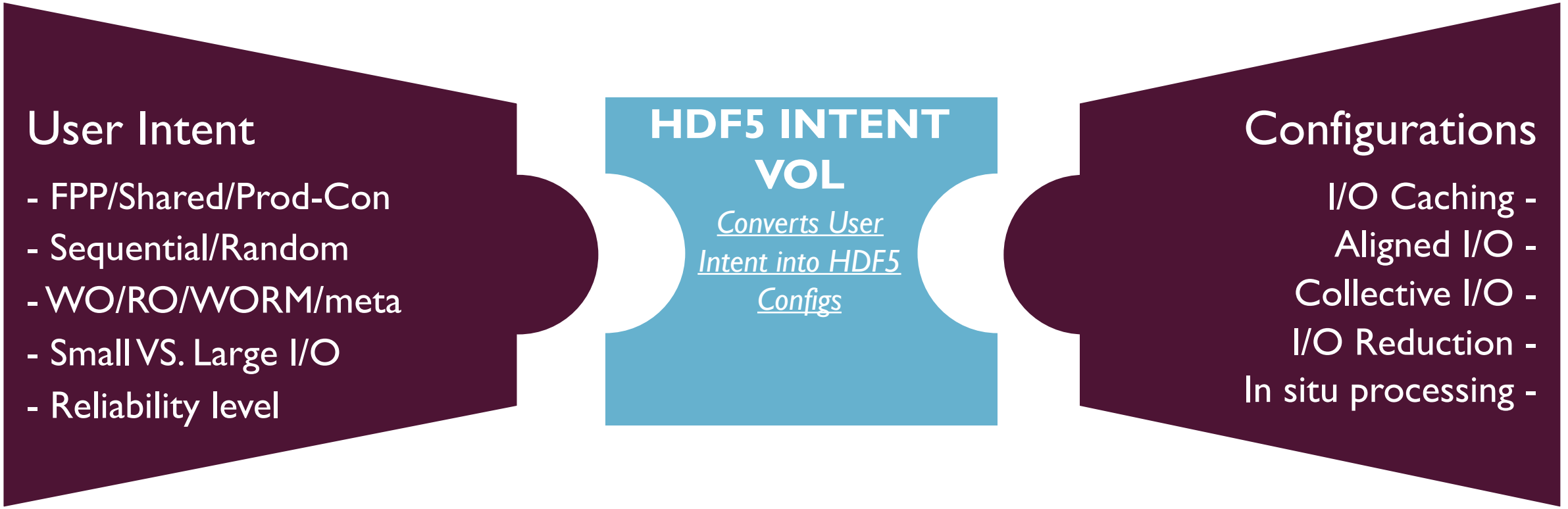


MISMATCH

Configurations

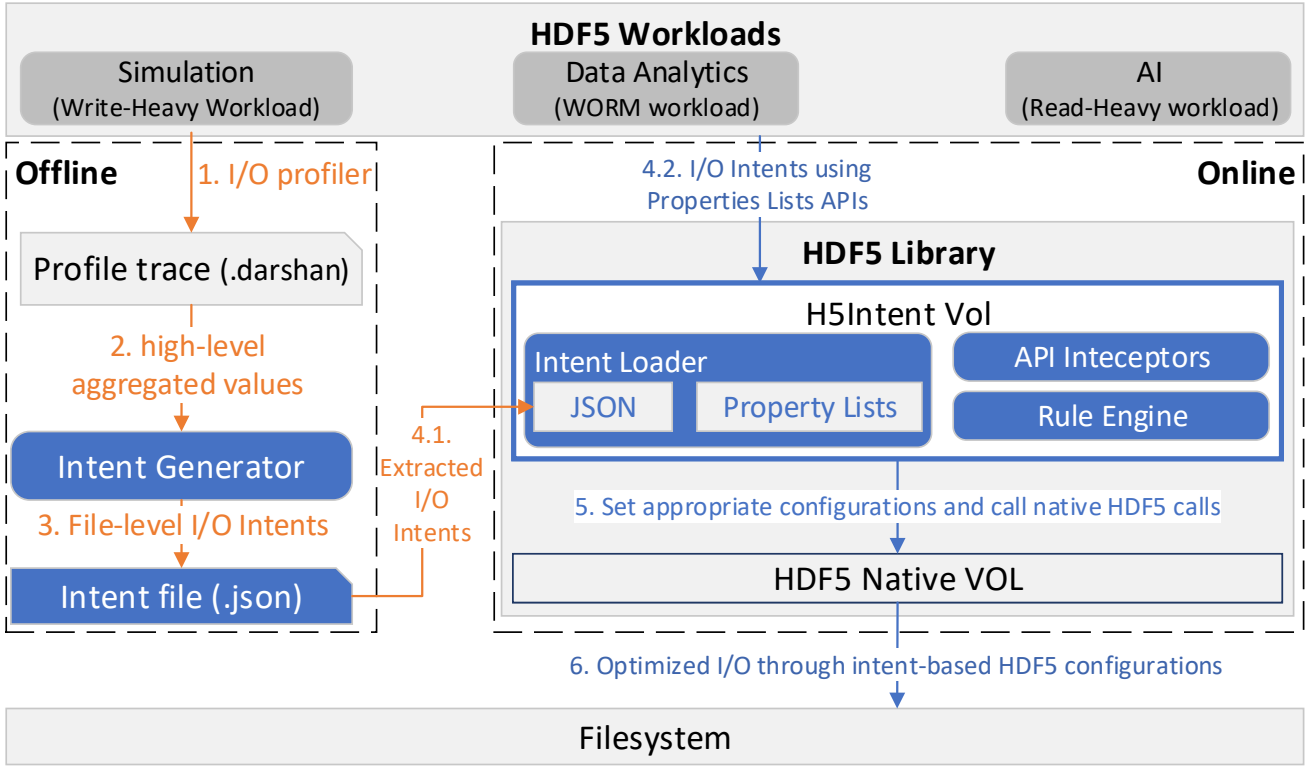
I/O Caching -
Aligned I/O -
Collective I/O -
I/O Reduction -
In situ processing -

IDEA: CHANGE IN PARADIGM



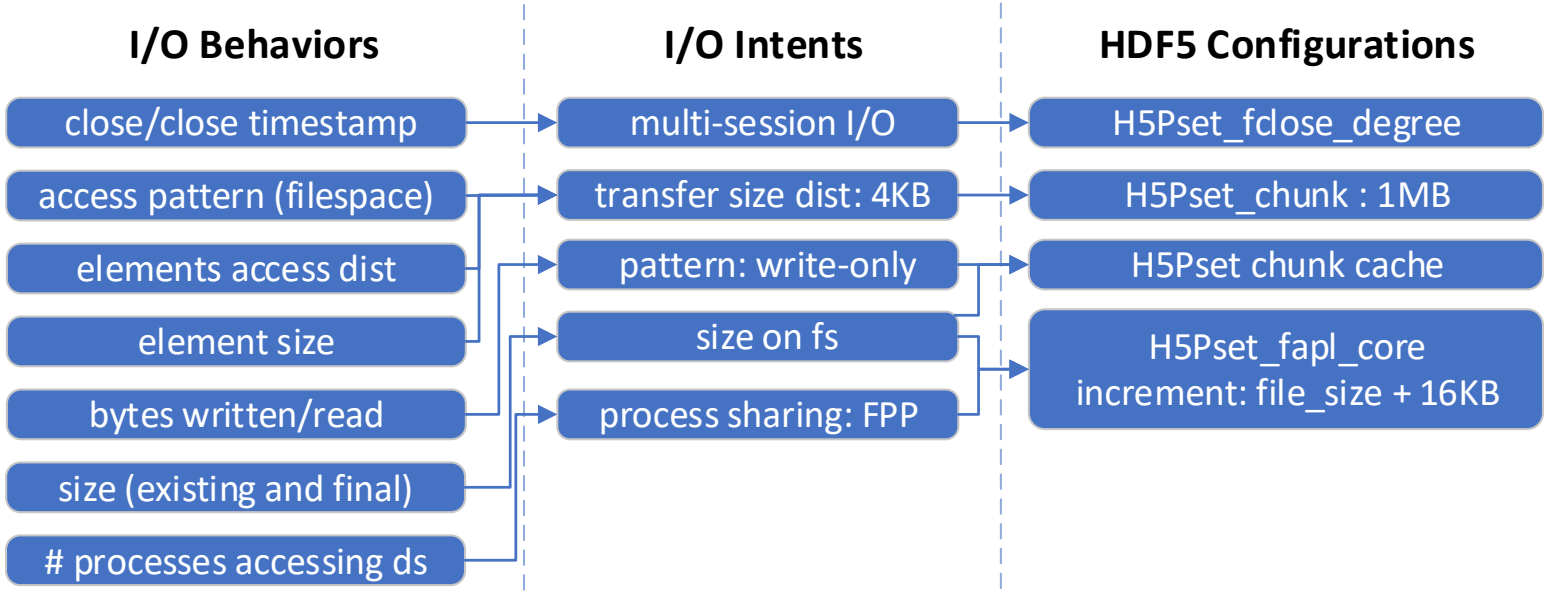
HIGH-LEVEL DESIGN

- Creation of Intents happens **offline**.
  - We use Darshan high-level profiling to generate I/O intents.
- Implementation of H5Intent is through the **VOL** *and* **property lists** infrastructure.
  - The property lists infrastructure is used for storing Intents for files and datasets
  - The VOL queries the properties and applies the correct optimizations at runtime.
- Rule engine contains the heuristic mapping of I/O intents into HDF5 configurations.



EXAMPLE OF USER INTENT FLOW.

- I/O behaviors are extracted from Darshan traces.
- IntentGenerator tool converts behaviors into I/O intents.
- Rule engine within the VOL applies different HDF5 configurations for the user.



H5INTENT WITH VPIC-IO ON LASSEN

Intents:

1. transfer size: 1MB and 256KB,
2. access pattern: write-only
3. the size on fs: 1.2TB
4. hot segments: sequential access on recency
5. process sharing: shared-file collective I/O

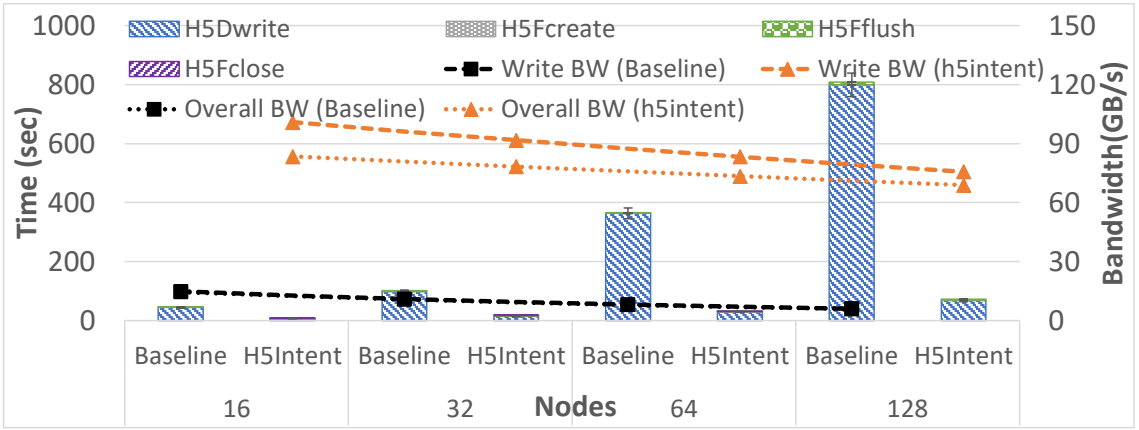
HDF5 File Configurations:

1. mpiio: comm-size
2. cache: size-1MB and blocks-5120,
3. close degree: STRONG

HDF5 Dataset Configurations:

1. chunk cache: size-1MB and slots-5120,
2. chunk: size-10GB,
3. mpiio chunk opt: multi,
4. mpiio chunk opt num: 40
5. mpiio chunk opt percent: 0.78%,
6. hyperslab selection: offset-0, size-256MB

VPIC



Result:

Bandwidth Improvement 8.3x-12.45x

IMPACT OF THIS WORK

HPC Users

- Better performance for their applications.
- Reduced complexity for configuring HDF5.
- Less error prone configurations, resulting in better I/O performance.
- Less training on new features of HDF5 and how to utilize them.

HDF5 Library

- Utilize Property List for describing Intent instead of configurations directly.
- More automatic application of new and existing features supported by HDF5.

Storage System

- Reduced risk of bad I/O due to misconfiguration of libraries.
- More predictable and stable I/O bandwidth for applications.

CODE DEMO WITH H5BENCH

RUNNING EXAMPLE
PROFILE THE APPLICATION

```
!./scripts/run_test.sh $case1_json $version 1
```

executed in 38.4s, finished 11:41:52 2023-07-06

```
Setting up darshan environment
Running up h5bench
Generated darshan file haridev_h5bench_write_id162345-162345_7-6-42098-76324319910067016 51_1.darshan
```

RUNNING CODE
GENERATE I/O INTENTS

```
!./scripts/generate_intents.sh $node $ppn $case1
```

executed in 10.2s, finished 11:42:20 2023-07-06

```
Running Intent Generator Tool
Generated intents file h5bench_write.json
```


```
⊖{
    "files": ⊖{
        "/p/gpfs1/haridev/temp/h5bench/sync-write-1d-s
trided-small_4_40/rank_0_test.h5": ⊖{
            "session_io": ⊕{4 items},
            "mode": 0,
            "process_sharing": ⊕[1 item],
            "fs_size": 1073741824,
            "sharing_pattern": 0,
            "ap_distribution": ⊕{4 items},
            "top_accessed_segments": {},
            "transfer_size_dist": ⊕{4 items},
            "ds_size_dist": ⊕{2 items}
        },
        "/p/gpfs1/haridev/temp/h5bench/sync-write-1d-s
trided-small_4_40/rank_1_test.h5": ⊕{9 items},
        "/p/gpfs1/haridev/temp/h5bench/sync-write-1d-s
trided-small_4_40/rank_2_test.h5": ⊕{9 items},
        "/p/gpfs1/haridev/temp/h5bench/sync-write-1d-s
trided-small_4_40/rank_3_test.h5": ⊕{9 items},
        "/p/gpfs1/haridev/temp/h5bench/sync-write-1d-s
trided-small_4_40/rank_4_test.h5": ⊕{9 items},
        "/p/gpfs1/haridev/temp/h5bench/sync-write-1d-s
trided-small_4_40/rank_5_test.h5": ⊕{9 items},
        "/p/gpfs1/haridev/temp/h5bench/sync-write-1d-s
trided-small_4_40/rank_6_test.h5": ⊕{9 items},
        "/p/gpfs1/haridev/temp/h5bench/sync-write-1d-s
```

RUNNING CODE
H5BENCH WITH WRITE-ONLY AND FPP

```
version=f"sync_none_none_{node}_{ppn}"
!./scripts/run_test.sh $case1_json $version 0
```

executed in 29.4s, finished 11:43:45 2023-07-06

```
Setting up environment
Running up h5bench
========= Performance Results =========
Total number of ranks: 160
Total emulated compute time: 0.000 s
Total write size: 160.000 GB
Raw write time: 1.886 s
Metadata time: 0.000 s
H5Fcreate() time: 0.045 s
H5Fflush() time: 0.924 s
H5Fclose() time: 0.005 s
Observed completion time: 3.610 s
SYNC Raw write rate: 84.812 GB/s
SYNC Observed write rate: 44.326 GB/s
```

```
version=f"sync_none_intent_{node}_{ppn}"
!./scripts/run_test.sh $case1_json $version 0
```

executed in 33.6s, finished 11:44:43 2023-07-06

```
Setting up environment
Running up h5bench
========= Performance Results =========
Total number of ranks: 160
Total emulated compute time: 0.000 s
Total write size: 160.000 GB
Raw write time: 0.376 s
Metadata time: 0.099 s
H5Fcreate() time: 1.688 s
H5Fflush() time: 2.103 s
H5Fclose() time: 1.981 s
Observed completion time: 9.485 s
SYNC Raw write rate: 425.215 GB/s
SYNC Observed write rate: 16.869 GB/s
```

Questions or Comments

Invitation to collaborate

Please find me to integrate your HDF5 optimizations with I/O Intents. We are happy to actively collaborate and help in integrating intent-driven optimizations for your solution.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC