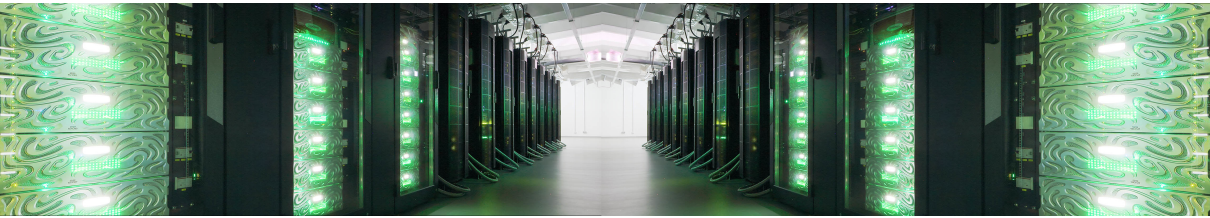
HPS
https://hps.vi4io.org

Prof. Dr. Julian M. Kunkel

IO500: Emerging Access Patterns and Features



Goals of the IO-500 Benchmarking Effort

- Foster paradigm shift from compute-centric to data-centric perspective
- Bound performance expectations for realistic workloads
- Track storage system characteristics behavior over the years
  - Foster understanding of storage performance development
  - Support to identify potent architectures for certain workloads
- Document and share best practices
  - Tuning of the system is encouraged
  - Submitters must submit detailed run parameters
- Support procurements, administrators and users

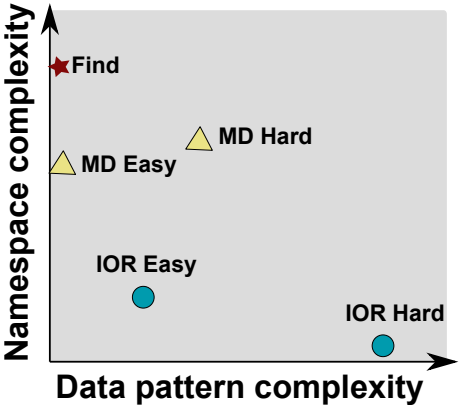
Disclaimer: While the author of this presentation is member of the IO500 committee and foundation, he speaks here his own opinion. The main purpose of the talk is to push the discussion and testing.

IO-500 Requirements

Requirements of the benchmarking

- **Representative**: for optimized or naive workloads
  - Describe the natural requirements for users
  - *-easy: upper bound for optimized * workload
  - *-hard: expected performance for non-optimized/naive applications
- **Inclusive**: cover various storage technology and non-POSIX APIs
  - At best: useful for HPC and Big Data workloads
- **Trustworthy**: representative results and prevent cheating
- **Cheap**: easy to run and short benchmarking time (in the order of minutes)

Covered Access Patterns



Old vision – figure from 2017

- IOR-easy: optimal (large-sequential) performance on (independent) files
- IOR-hard: small random performance on a shared file
- MD-easy: mdtest, per rank directory with empty files
- MD-hard: mdtest, shared directory with 3901 byte files
- find: query and filter files based on name, size and creation time

Benchmarking Phases in Standard Mode

1. Create
   1. IOR-easy write
   2. MD-easy create
   3. Timestamp (for find)
   4. IOR-hard write
   5. MD-hard create
2. Access (ranks shifted!)
   6. find files
   7. IOR-easy read
   8. MD-easy stat
   9. IOR-hard read
   10. MD-hard stat
3. Cleanup
   11. MD-easy delete
   12. MD-hard read, then delete

Stonewalling on writes/creates

- 300s independent write/create
- Wear-out: all procs catch up to fastest (largest size/file count)
- Simulates bulk-synchronous I/O

Issues

- Find searches all directories
- No concurrent usage of phases
- Lack of interactive MD usage
- No random patterns

Outline

IO500 Emerging Patterns

- IO500 supports standard mode (as before) and `-mode=extended`
- Runs more benchmark phases
- These are not used for scoring but they will need to be run for SC22
- The committee aims for justification documents for each new pattern
- Testing is welcome, though!

Benchmarking Phases in Extended Mode

1. Create
   1. IOR-easy write
   2. **IOR-rnd4k write**
   3. MD-easy create
   4. **IOR-rnd1M write**
   5. ***MD-Workbench create**
   6. *Timestamp (for find)
   7. **Find-easy**
   8. IOR-hard write
   9. MD-hard create
2. Access (ranks shifted!)
   10. find files
   11. **IOR-rnd4k read**
   12. **IOR-rnd1M read**
   13. **find hard**
3. *Access (continued)*
   10. **MD-Workbench bench**
   11. **!Concurrent**
   12. IOR-easy read
   13. MD-easy stat
   14. IOR-hard read
   15. MD-hard stat
4. Cleanup
   11. ***MD-Workbench cleanup**
   12. MD-easy delete
   13. MD-hard read
   14. MD-hard delete

- *Phase is not scored
- !GitHub PR for testing

Find Easy/Hard

- Find: measures both easy+hard dirs
  - Parallelizing search in shared directory is difficult
  - Ratio between easy/hard dirs influences score
  - Searches for size, timestamp and name
- Find-easy: search for name in MDTest-easy dirs
- Find-hard: search for size, timestamp, name in MDTest-hard

IOR-Rnd*

- Randomize access of a specific block size
- Pattern:
  - Block Size = 1 GiB * rank count
  - Assign randomly each record inside block to a rank
  - A rank repeats its random pattern for multiple blocks (segments)
  - Stonewall: all blocks are completely filled, except the last
  - Reason: Slow to create and store 1000 M random offsets...
- Option: ranks can prefill a block (collective operation), then move to next

MD-Workbench

- Simulates concurrent system usage
- Retain a fixed number of files
- Per iteration execute: stat, read, delete, write a single file (4x ops)

				Creation rate (creates/s)		
System	Nodes	PPN	D	Precreate	Bench T[1]=0	Bench T=4
ALCF Cooley (GPFS)	10	10	1	6,500	5,640	8,300
Düsseldorf (Lustre)	8	10	1	47,600	12,600	30,700
Düsseldorf (IME+Lustre)	8	10	1	4,500	1,550	4,460
DKRZ Mistral (Lustre)	10	10	1	21,800	2,380	2,220
KAUST (1 DataWarp BB)	10	10	1	3,800	3,390	14,600
KAUST (8 DataWarp BB)	10	10	1	25,600	8,190	32,000
NERSC (8 DataWarp BB)	10	10	1	19,000	8,560	35,100

J. Kunkel, G. Markomanolis. Understanding Metadata Latency with MDWorkbench

[1] Waiting time relative to execution time

Concurrent

- Runs multiple benchmarks concurrently
  - How does I/O write, read and metadata influence each other?
- Requires at least 5 procs
  - 1 runs IOR-easy write ($\leq 20\%$)
  - 2 run IOR-rnd-1M read ($\leq 60\%$)
  - 2 run MD-Workbench bench (rest)
- Calculate score using arithmetic mean
  - Weighted with actual proc counts

Results: IME - 64 Nodes, 2048 Procs

```
IO500 version io500-isc22_v1-11-gfb1b3608bc68 (extended)
[RESULT]       ior-easy-write       87.288724 GiB/s : time 312.630 seconds
[RESULT]    ior-rnd4K-write       11.537606 GiB/s : time 357.356 seconds
[RESULT]    mdtest-easy-write       30.518996 kIOPS : time 296.435 seconds [INVALID]
[RESULT]    ior-rnd1MB-write       61.666072 GiB/s : time 445.702 seconds
[      ]   mdworkbench-create       26.607830 kIOPS : time 69.770 seconds
[      ]            timestamp        0.000000 kIOPS : time 0.001 seconds
[RESULT]           find-easy       61.667290 kIOPS : time 146.148 seconds
[RESULT]       ior-hard-write       56.844883 GiB/s : time 331.302 seconds
[RESULT]    mdtest-hard-write        4.715099 kIOPS : time 328.578 seconds
[RESULT]                find        3.662263 kIOPS : time 3380.485 seconds
[RESULT]       ior-rnd4K-read        2.019697 GiB/s : time 2028.079 seconds
[RESULT]      ior-rnd1MB-read      100.700148 GiB/s : time 244.147 seconds
[RESULT]           find-hard        0.412095 kIOPS : time 3747.219 seconds
[RESULT]    mdworkbench-bench       57.329845 kIOPS : time 310.828 seconds
[RESULT]          concurrent     1693.992082 kIOPS : time 224.968 seconds [INVALID] <- output to be fixed
ior-easy-write = 43.228593b ior-rnd1MB-read = 43.693175  ior-md-workbench = 19.189264
[RESULT]        ior-easy-read      194.379764 GiB/s : time 140.394 seconds
[RESULT]     mdtest-easy-stat      102.165505 kIOPS : time 89.245 seconds
[RESULT]        ior-hard-read       44.434147 GiB/s : time 423.811 seconds
[RESULT]     mdtest-hard-stat       39.264197 kIOPS : time 40.372 seconds
[      ]   mdworkbench-delete       45.463460 kIOPS : time 40.888 seconds
[RESULT]   mdtest-easy-delete       41.715637 kIOPS : time 221.490 seconds
[RESULT]     mdtest-hard-read       53.832519 kIOPS : time 29.740 seconds
[RESULT]   mdtest-hard-delete        7.165160 kIOPS : time 222.349 seconds
[SCORE ] Bandwidth 80.910401 GiB/s : IOPS 20.723358 kiops : TOTAL 40.947958 [INVALID]
[SCOREX] Bandwidth 39.724822 GiB/s : IOPS 17.579657 kiops : TOTAL 105.761575 [INVALID]
```

Results: Lustre SSD Pool

```
IO500 version io500-isc22_v1-11-gfb1b3608bc68 (extended)
[RESULT]       ior-easy-write       29.309342 GiB/s : time 631.849 seconds
[RESULT]    ior-rnd4K-write        0.033010 GiB/s : time 480.300 seconds
[RESULT]    mdtest-easy-write       58.995556 kIOPS : time 382.948 seconds
[RESULT]     ior-rnd1MB-write        7.464711 GiB/s : time 538.108 seconds
[      ]   mdworkbench-create       64.761377 kIOPS : time 55.463 seconds
[      ]            timestamp        0.000000 kIOPS : time 0.001 seconds
[RESULT]           find-easy    29289.830225 kIOPS : time 0.790 seconds
[RESULT]    ior-hard-write        0.741015 GiB/s : time 363.107 seconds
[RESULT]    mdtest-hard-write       46.806830 kIOPS : time 386.433 seconds
[RESULT]                find     1093.302924 kIOPS : time 40.350 seconds
[RESULT]       ior-rnd4K-read        0.029269 GiB/s : time 541.786 seconds
[RESULT]     ior-rnd1MB-read       11.868263 GiB/s : time 336.853 seconds
[RESULT]           find-hard      698.548184 kIOPS : time 25.845 seconds
[RESULT]    mdworkbench-bench       95.919321 kIOPS : time 289.004 seconds
[RESULT]          concurrent      936.860721 kIOPS : time 189.365 seconds [INVALID]
ior-easy-write = 22.621297 ior-rnd1MB-read = 9.052912 ior-md-workbench = 29.938374
[RESULT]        ior-easy-read       38.256687 GiB/s : time 484.042 seconds
[RESULT]     mdtest-easy-stat      112.822177 kIOPS : time 200.729 seconds
[RESULT]        ior-hard-read       26.978562 GiB/s : time 10.258 seconds
[RESULT]     mdtest-hard-stat      106.913828 kIOPS : time 169.775 seconds
[      ]   mdworkbench-delete      120.680154 kIOPS : time 30.117 seconds
[RESULT]   mdtest-easy-delete       36.178402 kIOPS : time 624.284 seconds
[RESULT]     mdtest-hard-read       59.553531 kIOPS : time 303.946 seconds
[RESULT]   mdtest-hard-delete       33.446725 kIOPS : time 540.889 seconds
[SCORE ] Bandwidth 12.236010 GiB/s : IOPS 84.601095 kiops : TOTAL 32.174211
[SCOREX] Bandwidth 2.572622 GiB/s : IOPS 176.412659 kiops : TOTAL 75.195797 [INVALID]
```

Results: Lustre HDD Pool

```
IO500 version io500-isc22_v1-11-gfb1b3608bc68 (extended)
[RESULT]       ior-easy-write       54.110634 GiB/s : time 366.833 seconds
[RESULT]    ior-rnd4K-write        0.063898 GiB/s : time 442.268 seconds
[RESULT]    mdtest-easy-write      34.673054 kIOPS : time 325.948 seconds
[RESULT]    ior-rnd1MB-write        5.410805 GiB/s : time 400.131 seconds
[      ]  mdworkbench-create      38.765777 kIOPS : time 54.414 seconds
[      ]            timestamp        0.000000 kIOPS : time 0.000 seconds
[RESULT]           find-easy    20771.692986 kIOPS : time 0.564 seconds
[RESULT]    ior-hard-write        0.621490 GiB/s : time 433.024 seconds
[RESULT]    mdtest-hard-write      31.149733 kIOPS : time 462.196 seconds
[RESULT]                find     1125.447429 kIOPS : time 24.633 seconds
[RESULT]     ior-rnd4K-read        0.220360 GiB/s : time 128.489 seconds
[RESULT]    ior-rnd1MB-read       10.838720 GiB/s : time 188.972 seconds
[RESULT]           find-hard      664.819835 kIOPS : time 21.628 seconds
[RESULT]    mdworkbench-bench      87.498728 kIOPS : time 188.383 seconds
[RESULT]          concurrent      764.305767 kIOPS : time 186.263 seconds [INVALID]
ior-easy-write = 41.375729 ior-rnd1MB-read = 4.398179 ior-md-workbench = 18.293287
[RESULT]        ior-easy-read       24.915064 GiB/s : time 796.355 seconds
[RESULT]    mdtest-easy-stat      109.309728 kIOPS : time 104.089 seconds
[RESULT]       ior-hard-read        4.323702 GiB/s : time 62.321 seconds
[RESULT]    mdtest-hard-stat      105.503996 kIOPS : time 137.207 seconds
[      ]  mdworkbench-delete      119.337950 kIOPS : time 17.943 seconds
[RESULT]  mdtest-easy-delete       38.889409 kIOPS : time 290.953 seconds
[RESULT]    mdtest-hard-read       55.331934 kIOPS : time 260.654 seconds
[RESULT]  mdtest-hard-delete       31.871831 kIOPS : time 452.194 seconds
[SCORE ] Bandwidth 7.758158 GiB/s : IOPS 74.621538 kiops : TOTAL 24.060874
[SCOREX] Bandwidth 2.719485 GiB/s : IOPS 154.076201 kiops : TOTAL 68.416858 [INVALID]
```

Outline

Other Features (IOR and IO500)

- GPU Direct (available in repo)
  - IOR repo contains big chunk of code for it
  - IO500 can use option allocateBufferOnGPU
  - Pretty much untested feature – has been developed using self-developed mock-up of GPUDirect API
  - Help for testing/finalization is welcome
- Rename pattern for mdtest or md-workbench?
- Deadline execution?
  - Could stop execution, e.g., if wear-out exceeds $X \cdot 300s$
  - Some systems such as HDD based could benefit from fixed execution times
  - Could stop reading if exceeds $Y \cdot 300s$
- Recording runtime of individual operations
  - Similar to MD-Workbench?

Other Features: System Information

- The Comprehensive Data Center List (CDCL) of VI4IO and IO500 provides means to enter system information
  https://www.vi4io.org/io500-info-creator/
- Also tools are provided to automatically fill the schema!
- Need to enhance the tools and schema to match any storage system...

Discussion

- What is missing to make IO500 even more comprehensive?
- How to make it easier to use?

File Tree Changes During Benchmark Phase

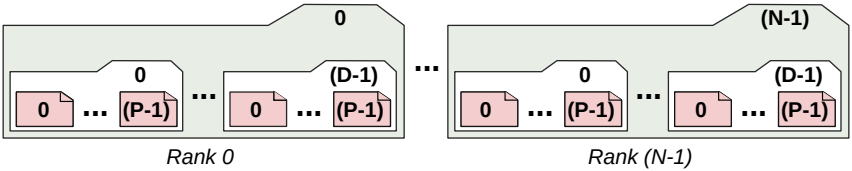


Figure: Working set (directory tree) after pre-creation phase

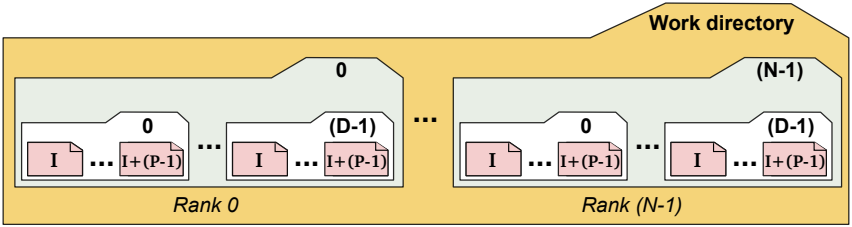


Figure: Working set (directory tree) after one iteration of the benchmark phase