HPS

https://github.com/IO500/submission-data

Julian Kunkel, Aasish Kumar Sharma, Anila Ghazanfar, Sepehr Mahmoodian, Sascha Safenreider

A Treasure Trove of Performance

Analyzing the IO500 Submission Data

Table of contents

Why look beyond the IO500 leaderboard?

Observation

- IO500 ranks HPC storage systems.
- Rankings focus on aggregate scores.
- Scores hide low-level behavior.

Untapped data source

- Per-process timings
- Stonewall statistics
- File close durations
- System metadata

Research questions

1. How variable are IO500 scores across public submissions?
2. Which phases are correlated, and which reveal independent behavior?
3. What does per-node and per-process normalization reveal about efficiency?
4. Which performance effects become visible only through log-level analysis?

Core claim

- IO500 submissions are more than benchmark results.
- They form a reusable dataset for storage-system analysis.

IO500 benchmark components

IOR: bandwidth

- **IOR-easy**: optimized file-per-process I/O pattern.
- **IOR-hard**: shared-file access with small interleaved records.

MDTest and pfind: metadata and namespace traversal

- **MDTest-easy**: each process creates files in its own directory.
- **MDTest-hard**: all processes create small files in one shared directory.
- **pfind**: parallel traversal of the file trees created by MDTest.

Composite scoring

Bandwidth score

$$Score_{BW} = \left(S_{ior\text{-}easy\text{-}w} \cdot S_{ior\text{-}easy\text{-}r} \cdot S_{ior\text{-}hard\text{-}w} \cdot S_{ior\text{-}hard\text{-}r}\right)^{1/4}$$

Metadata score

$$Score_{MD} = \left(S_{md\text{-}easy\text{-}w} \cdot S_{md\text{-}easy\text{-}s} \cdot S_{md\text{-}hard\text{-}w} \cdot S_{md\text{-}hard\text{-}s} \cdot S_{find}\right)^{1/5}$$

Overall score

$$Score_{overall} = \sqrt{Score_{BW} \times Score_{MD}}$$

- Final score uses the geometric mean of bandwidth and metadata.

Dataset analyzed

- **61 complete IO500 submission packages** from four competition lists:
  - ISC21, SC21, ISC22, and SC22
- Statistical methods used:
  1. Spearman rank correlations with false discovery rate correction
  2. Kruskal-Wallis tests for group comparisons

Dataset composition

File system	N	Interconnect	N
Lustre	27	IB HDR, 200 Gb/s	22
GPFS / Spectrum Scale	12	IB EDR, 100 Gb/s	18
DAOS	10	Omni-Path, 100 Gb/s	11
WekaFS	7	Other / Unknown	10
BeeGFS / Other	5		

The secret data inside IO500 submissions

Beyond leaderboard scores

Each IO500 submission package contains additional information that is usually not visible in the public ranking table.

- **Per-process CSV timings**: start/end times for benchmark operations.
- **Stonewall statistics**: 300-second limit and wear-down behavior.
- **Close-operation durations**: time spent closing files after I/O.
- **pfind traces**: work distribution during parallel namespace traversal.
- **System metadata**: file system, interconnect, nodes, and setup.

Why this matters

- Logs explain behavior hidden by rankings.

Finding 1: Score distributions are highly skewed

Score variability

- Overall scores span roughly **four orders of magnitude**.
- Metadata performance is more variable than bandwidth.
- Hard phases show the strongest variability.

Distribution effect

- A few large systems dominate the score distribution.
- Averages do not represent a typical submission.

Interpretation

- Interpret IO500 results at phase level, not only by averages.

Finding 2: Bandwidth and metadata cluster separately

Spearman rank correlations

- IOR phases correlate strongly: $r_s = 0.78$ to 0.96.
- MDTest phases cluster tightly: $r_s = 0.89$ to 0.98.
- Cross-domain correlations are weaker per process.

Composite-score effect

- ScoreBW and ScoreMD correlate strongly: $r_s = 0.92$.
- Pearson is lower: $r = 0.53$.
- Skew and outliers affect linear correlation.

Interpretation

- Correlations show structure, but not phase redundancy.

Finding 3: Per-node scores expose efficiency differences

- Raw leaderboard scores are dominated by total system scale.
- Per-node normalization shows architectural efficiency.
- Faster interconnects often show higher per-node scores.
- Strongest association appears for large sequential bandwidth.

Kruskal-Wallis results

- Overall per-node score: $H = 4.40$, $p = 0.111$, $\eta^2 = 0.11$.
- IOR-easy per-node bandwidth: $H = 11.16$, $p = 0.004$, $\eta^2 = 0.28$.

Caution

- Association, not causation: many system factors vary together.

Log insight 1: Close time is application-visible cost

Finding

- IOR logs expose time spent in file close operations.
- Close overhead differs clearly across file systems.
- Lustre shows visible overhead; DAOS shows very small overhead.
- Close time is included in IOR timing and affects applications.

Interpretation

- Storage systems place persistence costs in different I/O phases.
- Aggregate throughput hides this difference.

Log insight 2: straggler shape reflects architecture

WekaFS pattern

- Clustered slow processes.
- Storage-bucket contention.
- Slowdowns are correlated.

Lustre pattern

- Contiguous slow MPI ranks.
- Suggests OST-level hotspots.
- May indicate striping imbalance.

Main point

- Same aggregate score can hide different bottleneck shapes.
- Process-level logs show how load is distributed.

Log insight 3: parallel find remains load-imbalanced

- pfind can show extreme work skew.
- One process checks over **5M files**.
- Other processes check around **100K files**.
- MDTest-hard is hard to distribute efficiently.
- Job stealing reduces skew, but waiting remains.

Opportunity

- pfind logs can guide better traversal scheduling.
- They also help explain metadata-side scores.

Practical implications

For procurement

- Compare per-node and phase-level scores.
- Match phases to target workloads.

For benchmarking practice

- Inspect timing breakdowns.
- Check close time and stonewall behavior.
- Use per-process traces.

For the IO500 community

- Improve metadata quality.
- Report NIC count and storage targets.
- Include media type and tuning context.

Conclusion

The leaderboard ranks systems; the logs explain behavior.

- IO500 scores rank systems; logs explain behavior.
- Scores vary across architecture, scale, and tuning.
- Bandwidth and metadata show distinct patterns.
- Per-node/process views reveal hidden imbalance.
- Logs expose close costs, wear-down tails, and pfind skew.

Future work

- Analyze the full GitHub dataset.
- Study temporal trends and stragglers.
- Develop anomaly detection methods.