GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Georg-August-Universität Göttingen/GWDG

aasish-kumar.sharma@gwdg.de

Aasish Kumar Sharma

IO500 Benchmark Suite: Understanding and Running The Tests

High-Performance Computing System Administration (HPCSA)

Course Structure

Session Breakdown (90 minutes total)

→ Part 1: Theory (20-25 min) Benchmarking concepts, components, and metrics

→ Part 2: Theory (30-35 min) IO500 Benchmark concepts, components, and metrics

→ Part 3: Hands-on (35-40 min) Practical exercises on cluster

→ Part 4: Discussion (10-15 min) Q&A and troubleshooting

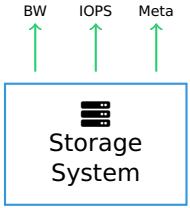
Learning Objectives

- By the end of this session, you will be able to:
  - Understand IO500 test components and scoring
  - Install and configure IO500 on HPC systems
  - Run benchmarks and interpret results
  - Diagnose common storage performance issues

Outline

What is IO500?

- **Industry standard** storage benchmark suite
- Measures **real-world** storage performance
- Used by supercomputing centers worldwide
- Provides **comparable** results across systems
- Tests **both** bandwidth and metadata



Real-World Analogy

- Traditional benchmarks are like testing a car's top speed on an empty highway.
- IO500 tests how it performs in real traffic, with turns, hills, and stops.

Why IO500 for HPC?

Research Computing Challenges

- Scientific workflows mix large data transfers with many small files
- Checkpointing requires consistent write performance
- Data analysis involves random access patterns
- Shared systems need fair resource allocation

Traditional Benchmarks:

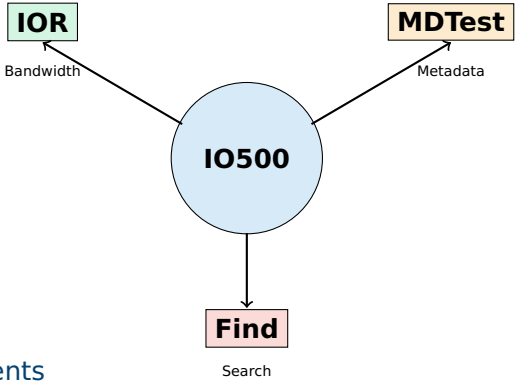
- Single workload
- Peak performance
- Limited applicability

IO500 Approach:

- Mixed workloads
- Sustained performance
- Real-world relevance

Outline

IO500 Test Suite



Three Main Components

- IOR - Measures data throughput (GiB/s)
- MDTest - Measures metadata operations (kIOPS)
- Find - Measures file discovery performance (kIOPS)

IOR: Interleaved Or Random Bandwidth Tests

Test	Access	Block	Workload
Easy Write	File-per-process	2 MiB	Checkpoint
Easy Read	File-per-process	2 MiB	Analysis
Hard Write	Shared file	47 KB	Logging
Hard Read	Shared file	47 KB	Database

Easy vs Hard

- **Easy:** Each process works independently (parallelizes well)
- **Hard:** Processes share files (tests coordination overhead)

MDTest: Metadata Operations

Operations Tested:

- File creation
- File stat (metadata query)
- File reading
- File deletion

Directory Structures:

- **Easy:** Flat (single level)
- **Hard:** Nested (tree)

Why This Matters for Research

- Version control systems (Git)
- Simulation output (millions of small files)
- Backup and archival operations
- Build systems and compilation

Outline

IO500 Scoring

Geometric Mean Formula

- Uses geometric mean to prevent good result from hiding
- One poor performance elsewhere could average the good one.

Bandwidth Score:

$$BW = \sqrt[4]{\prod IOR_{tests}}$$

IOPS Score:

$$IOPS = \sqrt[n]{\prod MD_{tests}}$$

Total Score:

$$IO500 = \sqrt{BW \times IOPS}$$

This ensures balanced performance across all workload types

Interpreting Results

Score Range	Classification	Typical System
> 10	High Performance	Top supercomputers
1–10	Good Performance	Research clusters
0.1–1	Entry Level	Department systems
< 0.1	Performance Issues	Needs attention

Performance Ratios to Watch

→ Easy/Hard Ratio - Reveals coordination overhead

→ Write/Read Ratio - Shows caching effectiveness

→ BW/IOPS Balance - Indicates system optimization focus

Outline

Quick Installation

Prerequisites

```
# Load MPI module
module load openmpi


# Verify MPI installation
mpicc --version

```

Build IO500

```
git clone https://github.com/IO500/io500.git

cd io500

git checkout io500-sc23
./prepare.sh

```

Basic Configuration

Minimal Configuration (config-test.ini)

```
[global]
datadir = ./datafiles
resultdir = ./results

[ior-easy]
transferSize = 2m
blockSize = 1g

[ior-hard]
transferSize = 47008
blockSize = 47008

[mdtest-easy]
files-per-proc = 10000

[mdtest-hard]
files-per-proc = 10000
tree-depth = 2

```

Note

- For learning,
  - Use smaller file counts and
  - Shorter runtimes

Outline

Running IO500

Single Node Test

```
# Run with 2 processes
mpirun -np 2 ./io500 config-test.ini


# Monitor progress
tail -f results/*/result.txt

```

Cluster Test (SLURM)

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --time=01:00:00

module load mpi
mpirun ./io500 config-cluster.ini
```

Reading Results

Example Output

```
[RESULT] ior-easy-write    0.142 GiB/s : time 383s
[RESULT] mdtest-easy-write 4.953 kIOPS : time 301s
[RESULT] ior-hard-write    0.105 GiB/s : time 323s
[RESULT] mdtest-hard-write 2.811 kIOPS : time 301s
[RESULT] find             40.483 kIOPS : time 58s
[SCORE ] Bandwidth 0.112 GiB/s : IOPS 7.505 kiops
[SCORE ] TOTAL 0.915 [INVALID]

```

Red Flags in This Output

- INVALID tag indicates data corruption
- Very low bandwidth (< 0.15 GiB/s)
- Good metadata performance (bright spot)

Outline

Exercise Session Overview

Setup (5 minutes)

- Connect to cluster
- Verify MPI environment
- Clone IO500 repository

Four Exercises (30 minutes)

1. **(10 min)** Installation and basic configuration
2. **(10 min)** Running your first benchmark
3. **(5 min)** Analyzing results
4. **(5 min)** Performance comparison

Wrap-up (5 minutes)

- Discussion of results
- Q&A

Exercise 1: Installation (10 min)

Objective

- Install IO500 on the cluster and verify the build

Tasks

1. SSH to cluster: `ssh username@cluster.uni-goettingen.de`
2. Load MPI module: `module load openmpi`
3. Clone repository: `git clone https://github.com/IO500/io500.git`
4. Checkout version: `cd io500 && git checkout io500-sc23`
5. Build: `./prepare.sh`
6. Verify: `ls -l io500`

Expected Output

- You should see compiled binaries: `io500`, `ior`, `mdtest`, `pfind`

Exercise 2: First Benchmark (10 min)

Objective

- Run a minimal IO500 test and observe execution

Tasks

```
# 1. Create test configuration
cp config-minimal.ini config-exercise.ini

# 2. Edit config (reduce file counts for speed)
vim config-exercise.ini
# Change files-per-proc to 1000

# 3. Run benchmark
mpirun -np 2 ./io500 config-exercise.ini

# 4. Monitor in another terminal
tail -f results/*/result.txt

```

Questions to Consider

- Which test takes longest? Why?
- What resources are being used?

Exercise 3: Result Analysis (5 min)

Objective

- Interpret your benchmark results and identify patterns

Tasks

```
# 1. View summary
cat results/*/result_summary.txt

# 2. Calculate Easy/Hard ratios
# Easy_Write / Hard_Write = ?
# Easy_Read / Hard_Read = ?

# 3. Identify issues
# Are any results marked INVALID?
# Which tests performed best/worst?

```

Group Discussion

- Compare your results with neighbors.
  - What differences do you see?
  - Why might they differ?

Exercise 4: Performance Comparison (5 min)

Objective

- Test with different configurations and compare

Tasks

```
# 1. Run with different process counts
mpirun -np 4 ./io500 config-exercise.
    ↪ ini

# 2. Compare results
diff results/run1/result_summary.txt \
     results/run2/result_summary.txt

# 3. Analyze scaling
# Does 2x processes = 2x performance?

```

Expected Observation

- Performance rarely scales linearly.
  - Why? (overhead, contention, limits)

Exercise Troubleshooting

Common Issues

→ Build fails - Check MPI module: `module list`, try `module load mpi`

→ Permission denied - Ensure write access: `chmod 755 datafiles`

→ Slow execution - Normal for complete runs; reduce file counts for testing

→ INVALID results - May indicate storage issues; check `df -h`

Get Help

- Check log files in `results/*/*.log`
- Consult with respective instructors
- Review documentation at `https://io500.org`

Outline

Key Takeaways

Understanding IO500

- Comprehensive benchmark suite for storage systems
- Tests both bandwidth (IOR) and metadata (MDTest)
- Uses geometric mean for balanced scoring
- Reflects real-world application behavior

Practical Skills Gained

- Installation and configuration
- Running benchmarks on HPC systems
- Interpreting performance results
- Identifying storage bottlenecks

Resources for Further Learning

Official Resources

- **Website:** https://io500.org
- **GitHub:** https://github.com/IO500/io500
- **Documentation:** https://io500.org/documentation
- **Results Database:** https://io500.org/list

Next Steps

- Run IO500 on your research filesystem
- Compare with other systems in IO500 list
- Experiment with different configurations
- Use results to guide storage optimizations

Questions & Discussion

Learn and Share!

Questions & Discussion

Feedback - Contact:
Aasish Kumar Sharma
aasish-kumar.sharma@gwdg.de

Materials:
https://hps.vi4io.org/teaching/ressources/start

Reference: Complete Config File

```
[global]
datadir = ./datafiles
resultdir = ./results

[debug]
stonewall-time = 300

[ior-easy]
transferSize = 2m
blockSize = 9920000m
API = POSIX

[ior-hard]
transferSize = 47008
blockSize = 47008
API = POSIX
```

```
[mdtest-easy]
files-per-proc = 1000000
API = POSIX

[mdtest-hard]
files-per-proc = 1000000
tree-depth = 3
branch-factor = 1
API = POSIX


# Find test
#===========

[find]
external-script = ./pfind
```

Reference: Monitoring Script

```
#!/bin/bash
# monitor-io500.sh - Real-time monitoring

while true; do
    clear
    echo "=== IO500 Monitor $(date) ==="
    echo
    echo "Disk Usage:"
    df -h ./datafiles | tail -1
    echo
    echo "I/O Statistics:"
    iostat -x 1 1 | grep -A5 "Device"
    echo
    echo "Active Processes:"
    ps aux | grep -E '(io500|ior|mdtest)' | grep -v grep
    sleep 10
done
```

Reference: Performance Analysis Checklist

When Analyzing Results

1. Check for INVALID markers (data integrity issues)
2. Calculate Easy/Hard ratios (coordination overhead)
3. Compare Write/Read performance (caching effects)
4. Examine BW vs IOPS balance (system optimization)
5. Look at individual test durations (bottlenecks)
6. Compare with similar systems in IO500 list

Red Flags

- Easy/Hard ratio > 3: Poor shared access handling
- Write/Read ratio > 2: Cache or read path issues
- Any INVALID results: Storage reliability concerns
- Extremely low scores: Configuration or hardware problems