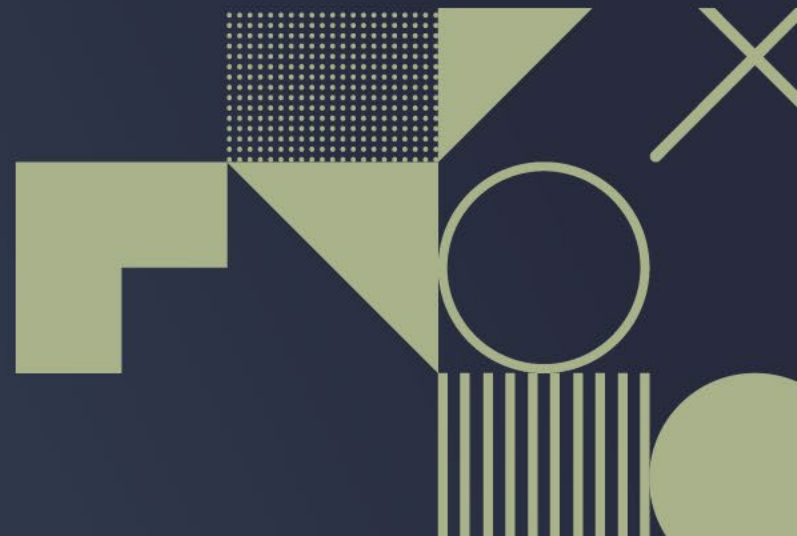
User-Centric System Fault Identification Using IO500 Benchmark

Radita Liem[*], Dmytro Povaliaiev[*], Julian Kunkel[‡], Jay Lofstead[†], Christian Terboven[*]

[*]Chair for High Performance Computing, IT Center, RWTH Aachen University
[‡]Göttingen University/GDWG
[†]Sandia National Laboratories



Introduction

I/O is a shared resource in a cluster. It creates performance variability on the running applications and makes predicting I/O performance difficult.

User often **relies on their own perception** that the application is slowing down[1] and the **current solutions mostly focused on system maintainer**.

We want to provide users with **easy-to-digest realistic expectations** about their application's I/O in the specific environment where their applications are running.

In this pursuit, **we propose to use the IO500 benchmark** as a way to manage user expectation for a HPC system.

[1]J. Kunkel and E. Betke, "Tracking user-perceived i/o slowdown via probing," inHigh Performance Computing, M. Weiland, G. Juckeland,S. Alam, and H. Jagode, Eds. Cham: Springer International Publishing,2019, pp. 169–182.

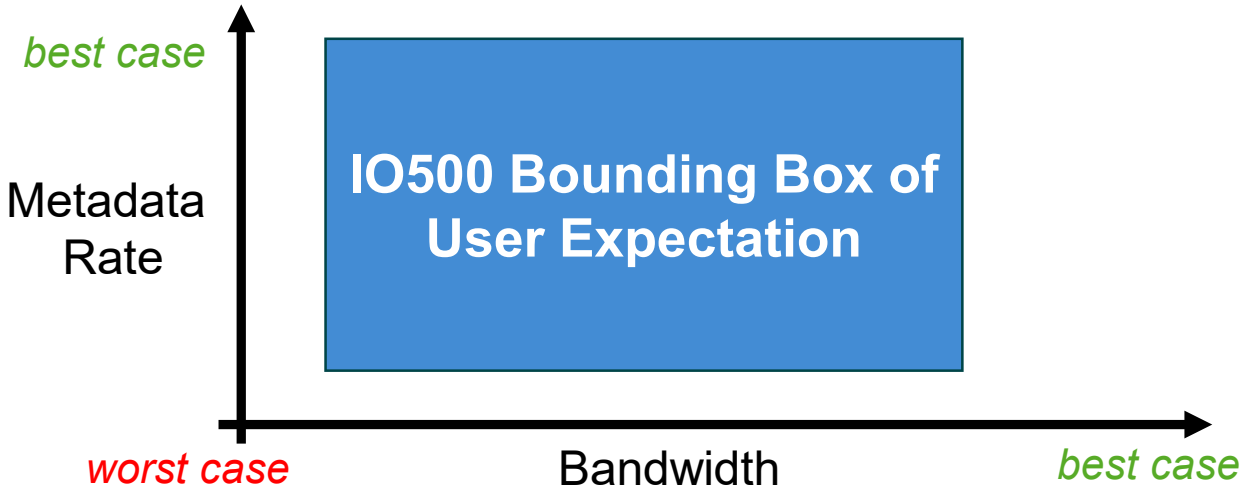
IO500 Benchmark Overview

- Developed in 2017 and quickly becoming the de facto benchmarking standard for HPC Storage[2].
- Designed to create balanced performance measurement
- The benchmark has 5 scenarios[3] :
  - **IOR 'easy':** Free to tune IOR parameters. Typically file-per-process, large, aligned chunks. Creating **best possible bandwidth performance**.
  - **IOR 'hard':** Limited options to tune. Forced to use small unaligned I/O to a single shared file. Providing **worst possible bandwidth performance**.
  - **mdtest 'easy':** Free to tune mdtest parameters with zero size files in separate directory per process. Creating **best case scenario for metadata rate**
  - **mdtest 'hard':** Limited options to tune. Forced all processes to write into a single shared directory. Providing **worst case scenario for metadata rate**.
  - **Find:** Finding specific subset of files created by 4 other benchmarks.
- The numbers from the scenarios are combined using geometric mean and each scenario is running for 300s that eliminates the cache effect

[2] J. Kunkel, "Virtual Institute for I/O," *ISC'19*: The IO-500 and the Virtual Institute of I/O, 30-Oct-2020. [Online]. Available: https://www.vi4io.org/io500/bofs/isc19/start. [Accessed: 27-May-2021].
[3] M. Rásó-Barnett, "Lustre and IO-500: Experiences with the Cambridge Data Accelerator", 2019. [Online]. https://www.eofs.eu/_media/events/lad19/03_matt_raso-barnett-io500-cambridge.pdf. [Accessed: 02-Mar-2021]

IO500 Benchmark Usage

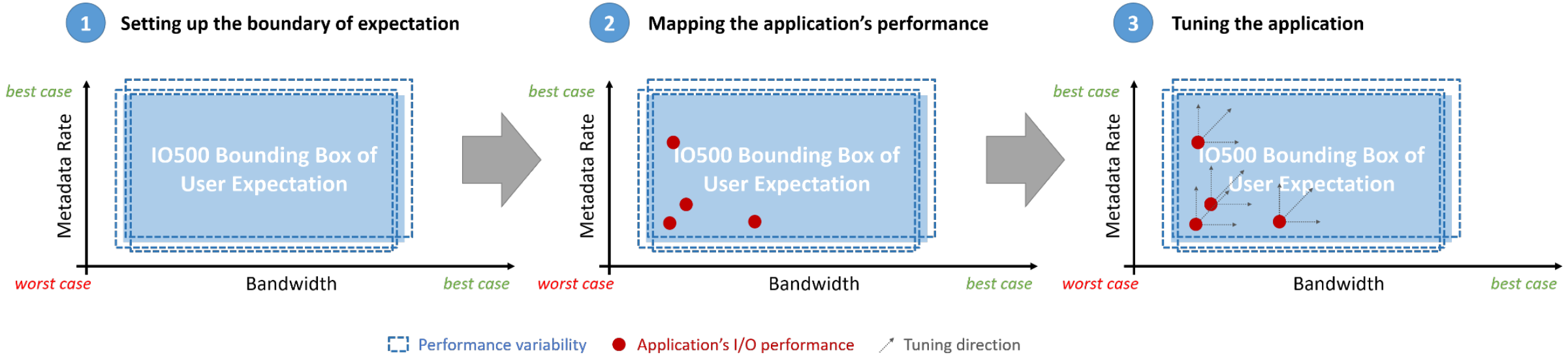
- IO500 benchmark's mdtest and IOR scenario can be used to form a bounding box of user expectations [4] as illustrated by the figure below



- **Worst case scenario** is coming from IOR and mdtest **'hard'** scenario
  Best case scenario is coming from IOR and mdtest '**easy**' scenario
- **'Find'** is not used in this bounding box model since it is not as controlled as IOR and mdtest and will skew the IO500 numbers

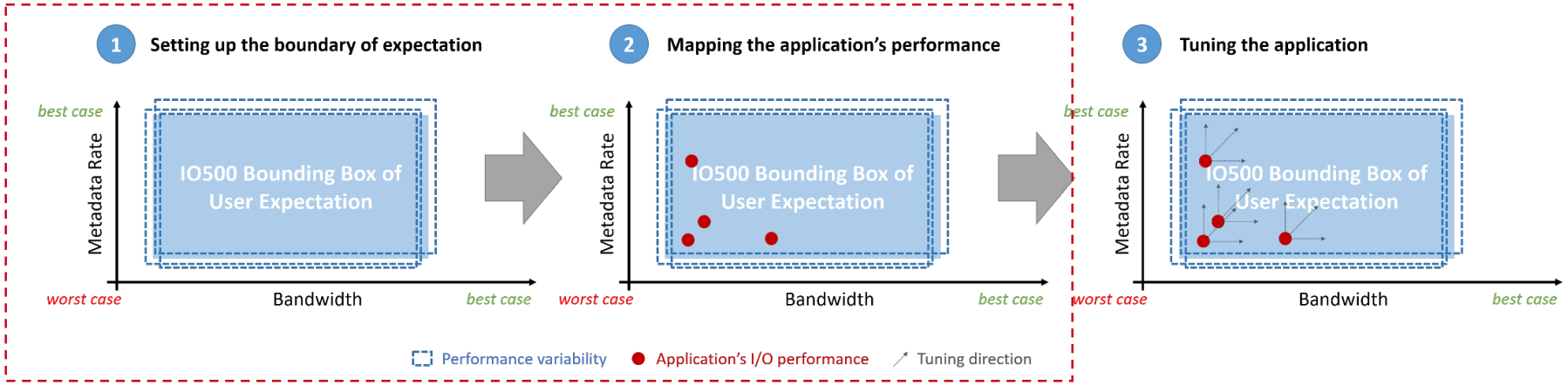
[4] A. Dilger, "IO500 | A storage Benchmark for HPC", 2019. [Online]. Available:https://wiki.lustre.org/images/9/92/LUG2019-IO500_Storage_Benchmark_for_HPC-Dilger.pdf. [Accessed: 02-Mar-2021]

IO500-based Workflow Proposed



Proof of Concept Experiment Setup

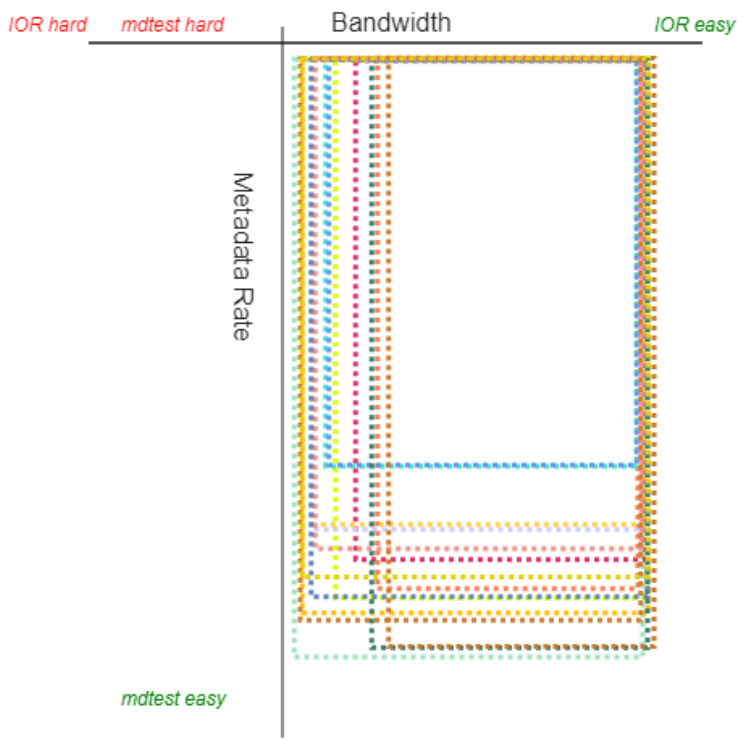
- The experimentation in this work covers **the first and second step of the workflow**. The second step of the work flow is still in our **exploratory stage** and the third step is for the future work



- **Experiment environment:**
  - CLAIX-2018 cluster at RWTH Aachen University (48 cores Intel Skylake, 384 GB memory), 40 Gb/s Ethernet.
  - 4 nodes BeeGFS Filesystem, each with 480 GB SSD.
  - IO500 benchmark - SC20 submission version.
  - NAS Parallel Benchmark – BTIO "full" class A,B, and C on 4,9, and 16 processes

Results: Forming Bounding Box of User Expectation [1]

- Bounding box of **POSIX** API, each square represents individual run from the same IO configuration

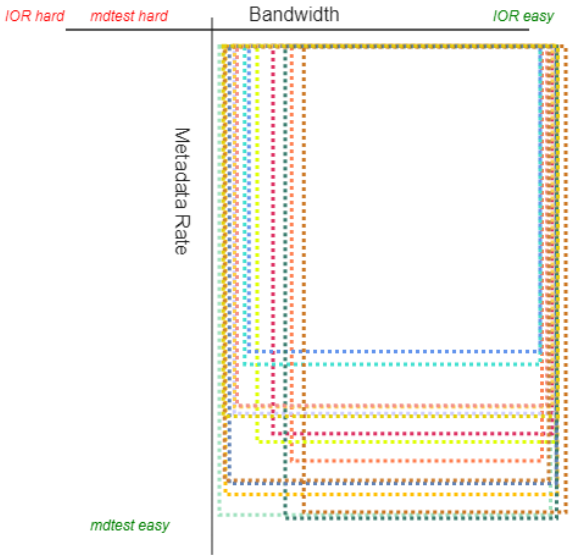


	IOR hard (GiB/s)	IOR easy (GiB/s)	mdtest hard (KIOPS)	mdtest easy (KIOPS)
	0.85	1.88	10.97	145.17
	1.03	1.89	11.26	123.26
	1.10	1.87	10.59	129.86
	0.87	1.90	10.76	135.27
	0.97	1.90	10.64	131.93
	0.94	1.86	11.06	102.35
	0.95	1.86	10.84	102.06
	0.90	1.89	10.98	116.54
	0.90	1.89	11.16	115.40
	1.08	1.90	11.14	143.09
	0.91	1.88	10.80	120.86
	0.90	1.90	11.05	131.65

This project is currently displayed in:https://bit.ly/3BhhAFZ

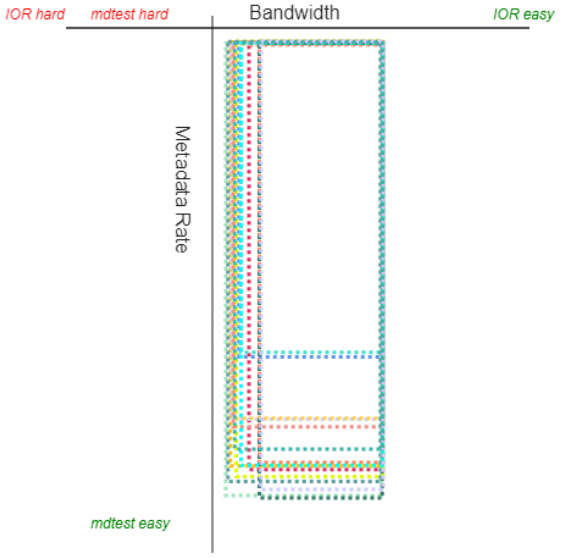
Results: Forming Bounding Box of User Expectation [2]

- Bounding box of **POSIX** API, read and write show different pattern



	IOR hard (GiB/s)	IOR easy (GiB/s)	mdtest hard (KIOPS)	mdtest easy (KIOPS)
	0.52	1.75	2.60	76.12
	0.72	1.76	2.56	63.37
	0.79	1.72	2.55	67.64
	0.55	1.78	2.62	72.92
	0.66	1.77	2.56	64.69
	0.62	1.71	2.57	52.50
	0.63	1.71	2.56	50.49
	0.58	1.77	2.54	60.23
	0.58	1.76	2.50	59.20
	0.77	1.77	2.64	76.67

Bounding box from POSIX **write** result



	IOR hard (GiB/s)	IOR easy (GiB/s)	mdtest hard (KIOPS)	mdtest easy (KIOPS)
	1.37	2.02	24.80	360.31
	1.47	2.03	26.02	341.16
	1.51	2.03	23.60	335.88
	1.39	2.03	24.04	338.53
	1.42	2.03	23.63	346.27
	1.44	2.03	24.96	254.26
	1.43	2.03	24.70	257.68
	1.40	2.03	25.13	303.84
	1.41	2.02	25.57	303.20
	1.51	2.02	25.22	362.23

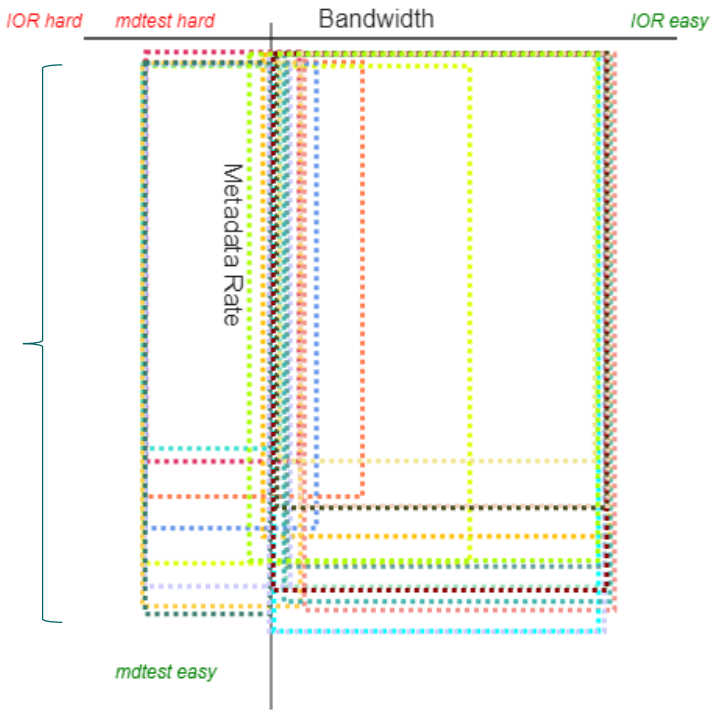
Bounding box from POSIX **read** result

This project is currently displayed in:https://bit.ly/3BhhAFZ

Results: Anomalous Bounding Box

- Anomalous result in **MPI-IO** API: IOR ‘Easy’ score gets lower number than IOR ‘hard’
- Broken node is most likely the reason behind these anomalous result

Bounding box skewed
to the direction of IOR ‘hard’



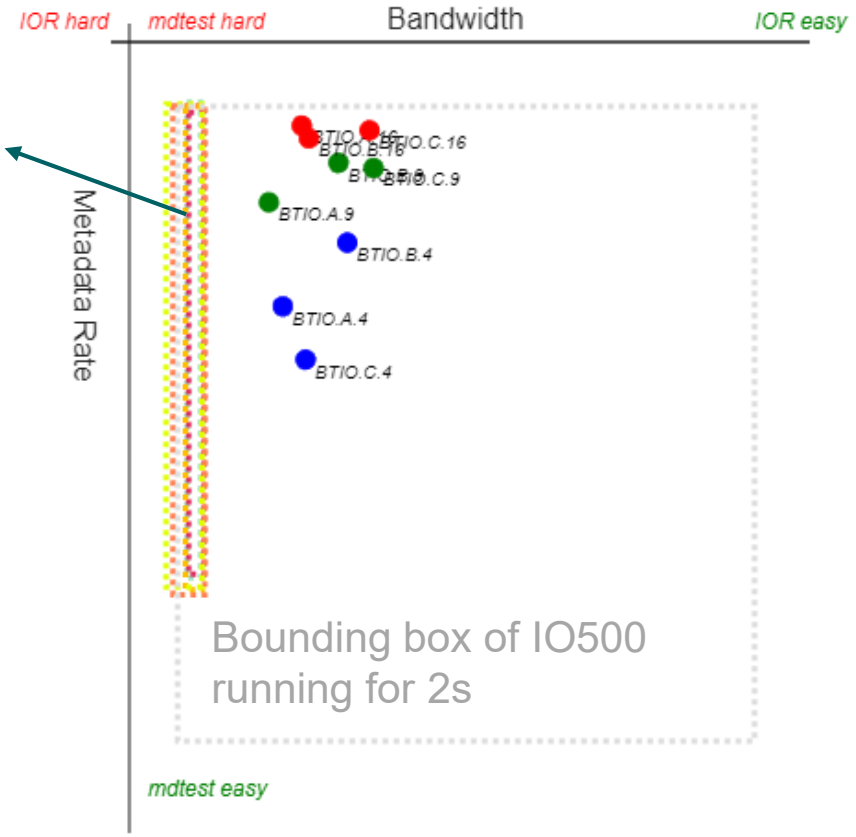
	IOR hard (GiB/s)	IOR easy (GiB/s)	mdtest hard (KIOPS)	mdtest easy (KIOPS)
	0.90	1.89	10.74	135.79
	0.94	0.47	10.50	106.45
	1.14	0.47	12.86	114.73
	0.83	1.89	11.12	124.06
	1.46	0.47	13.78	130.29
	0.88	0.47	13.50	103.47
	0.99	0.47	13.24	122.10
	0.91	0.47	13.04	135.73
	0.95	0.46	13.10	140.41
	0.85	0.47	13.02	142.20
	0.96	1.90	11.00	141.28
	0.86	1.85	10.81	146.24

This project is currently displayed in:https://bit.ly/3BhhAFZ

Results: Exploration on the I/O Performance Mapping [1]

- Exploration with BTIO shows the application's performance falls within the box for **MPI-IO API *with cache effect***

Bounding boxes of IO500 running with default setup (300s)

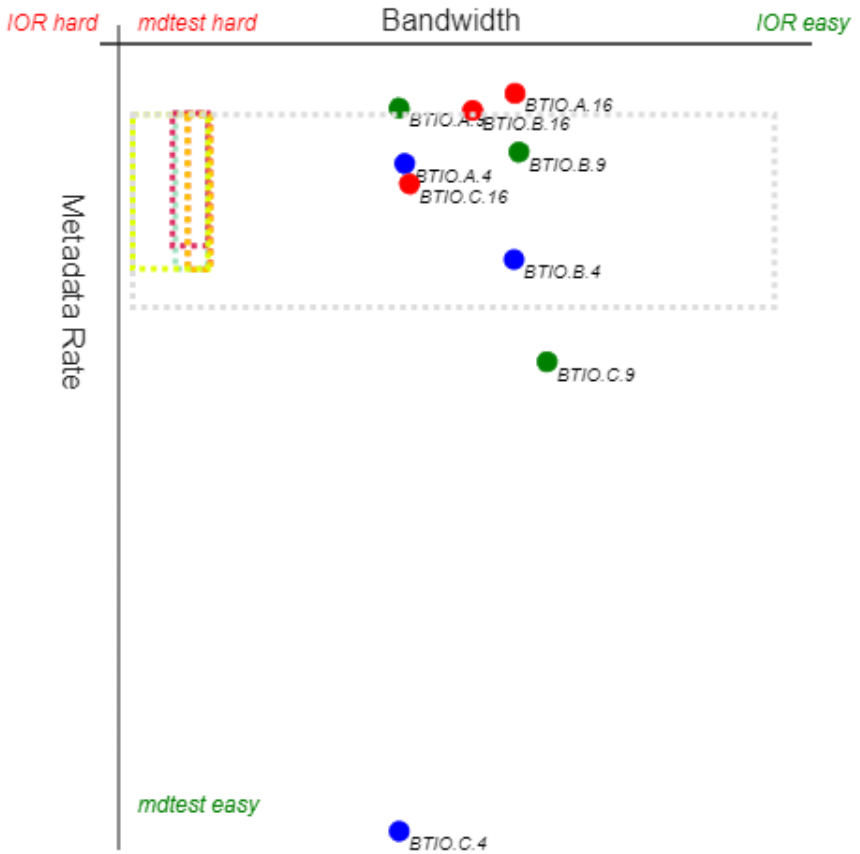


	Benchmark	Bandwidth (GiB/s)	Metadata Rate (KIOPS)
	BTIO.A.4	0.68	21.67
	BTIO.A.9	0.64	14.18
	BTIO.A.16	0.73	8.66
	BTIO.B.4	0.85	17.08
	BTIO.B.9	0.82	11.35
	BTIO.B.16	0.75	9.58
	BTIO.C.4	0.74	25.50
	BTIO.C.9	0.91	11.71
	BTIO.C.16	0.90	9.00

This project is currently displayed in:https://bit.ly/3BhhAFZ

Results: Exploration on the I/O Performance Mapping [2]

- However in the **POSIX API**, metadata rate calculation falls outside the cache box

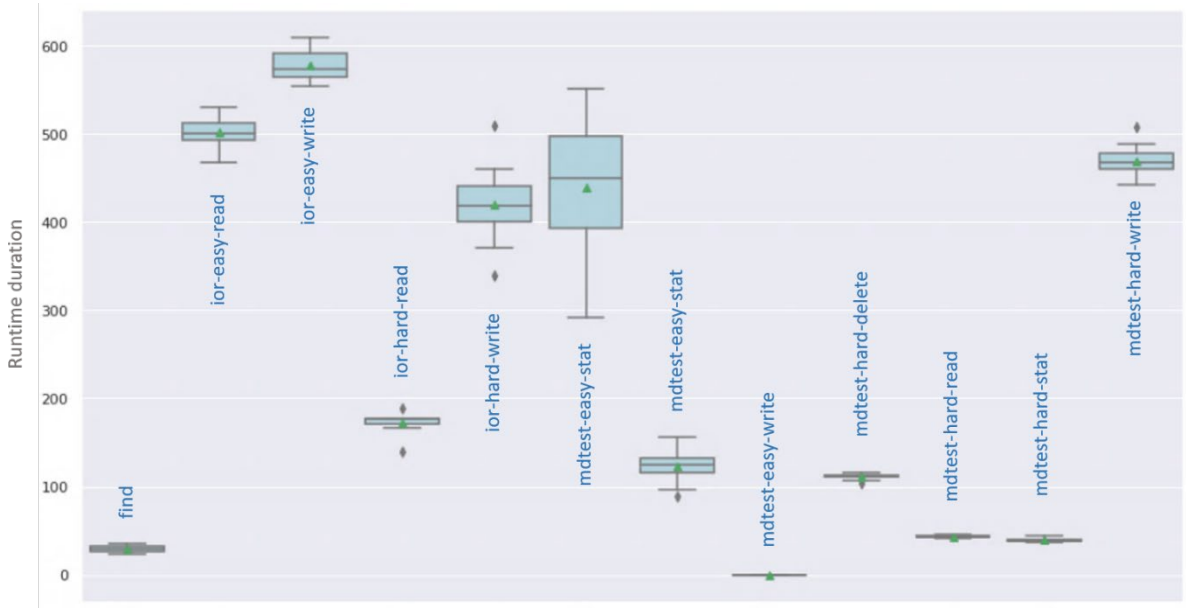


Benchmark	Bandwidth (GiB/s)	Metadata Rate (KIOPS)
BTIO.A.4	0.99	18.42
BTIO.A.9	0.97	5.99
BTIO.A.16	1.27	2.69
BTIO.B.4	1.27	39.90
BTIO.B.9	1.28	15.84
BTIO.B.16	1.16	6.51
BTIO.C.4	0.97	167.96
BTIO.C.9	1.36	62.81
BTIO.C.16	1.00	22.99

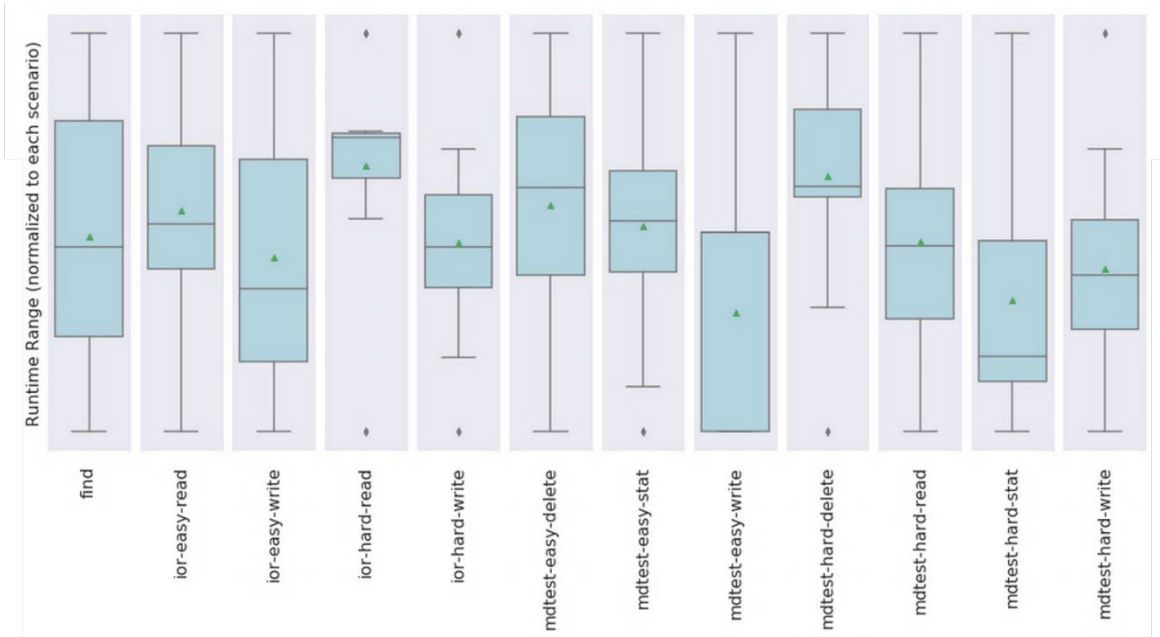
This project is currently displayed in:https://bit.ly/3BhhAFZ

Results: Tail Latency Observations

- Tail latency is one of the issue in the IO500 benchmark community forums with discussions about the impact of the stonewall requirement.



Runtime variety of each IO 500 scenario.
Several tests have significant variability



Close up look on the runtime range for each scenario.

Discussion and Future Work

- **The workflow provides a way to identify system fault**.
  - It provides sanity check using the IO500 ‘hard’ and ‘easy’ relationship.
  - Without users informed about this anomaly, they will just assume that the machine is operating slow but intended.
- The workflow shows a promising results and also **several tasks for future work**:
  - Refining the IO500 configuration
  - Testing on different filesystems, cluster, and APIs such as HDF5
  - Scaling up the experiment
- The exploratory work shows **the impact of cache effect** that needs further investigation and additional works on **exploring I/O tools and metrics**

Thank you!

Inquiry and question: **Radita Liem** (liem@itc.rwth-aachen.de)