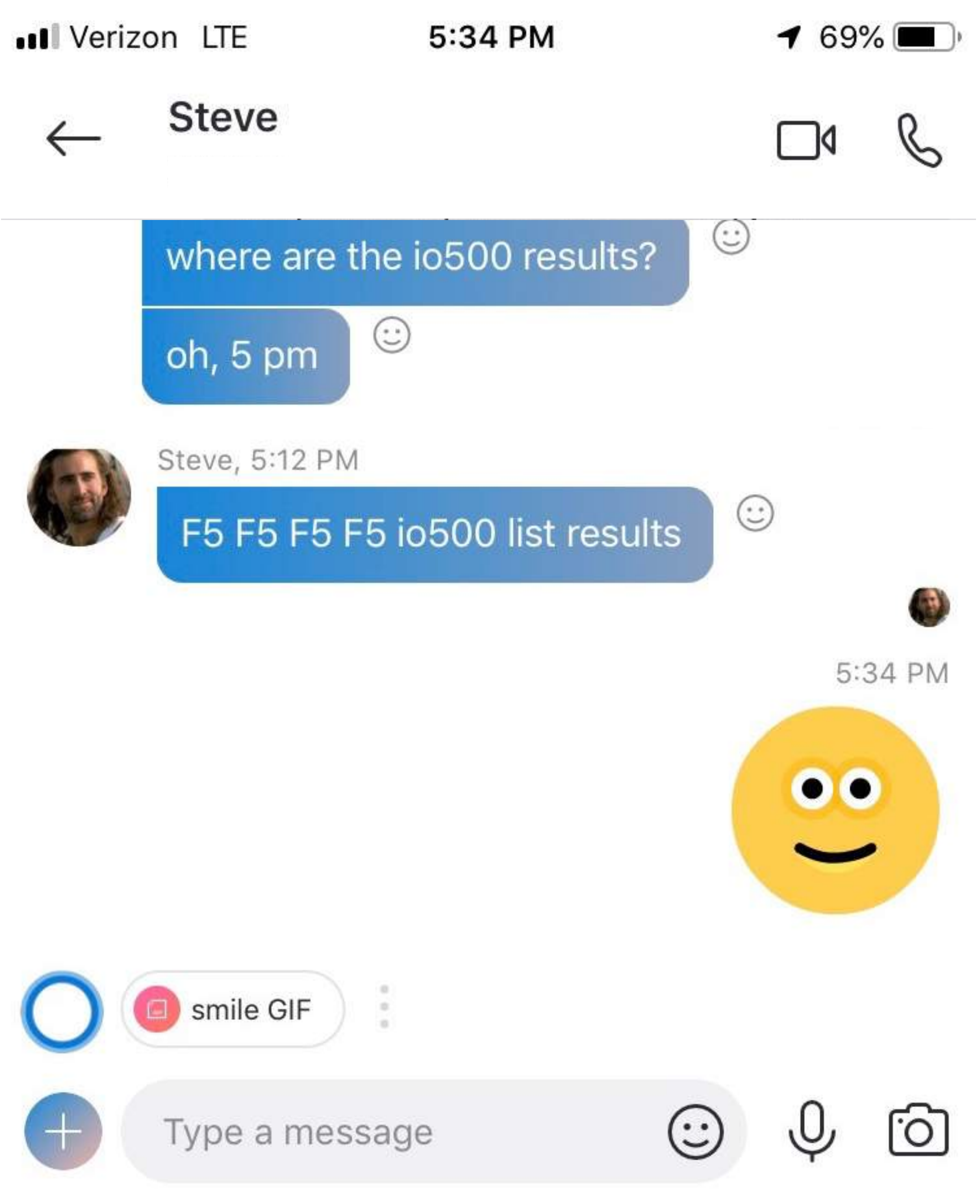
IO500 @ SC18

Bent, Lofstead, Kunkel, Markomanolis

IO500


Verizon LTE
5:34 PM
69%
Steve
where are the io500 results?
oh, 5 pm
Steve, 5:12 PM
F5 F5 F5 F5 io500 list results
5:34 PM
smile GIF
Type a message

Why IOR Hard Write is Difficult

```
read                                   |     write
disk I/O size   ios    % cum %         |  ios   % cum %
4K:             13977 100 100         |     6   0    0
8K:                 0   0 100         |     4   0    0
16K:                0   0 100         |    18   0    0
32K:                0   0 100         |    24   0    0
64K:                0   0 100         | 12999  96   96
128K:               0   0 100         |   406   3   99
256K:               0   0 100         |    68   0   99
512K:               0   0 100         |     3   0  100
```

This data is a histogram of IO sizes from an Lustre OST during the IOR hard test. Because each write is 47K, each will incur a 4K read due to read-modify-write of that page followed by a 48K write (which shows up in the 64K bucket).

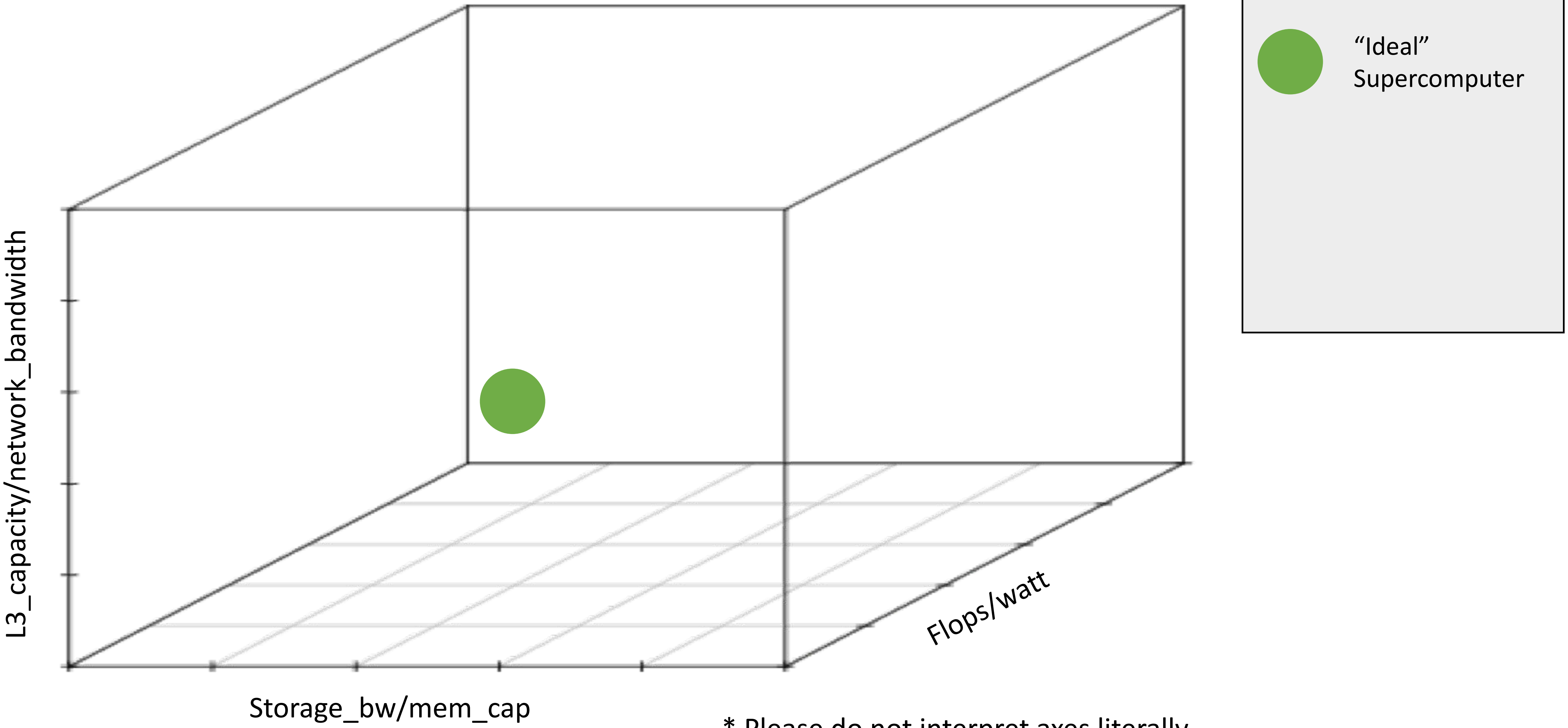
IO500 | Motivation: "How Fast Does a Disk Drive Go?"

- Firmware engineer: "150 – 160 MB/s"
- Marketer: "200 MB/s"
- Performance engineer: "130 MB/s"
- Salesperson: "100 MB/s"

- User: "Why do I only see 10 MB/s?!?"

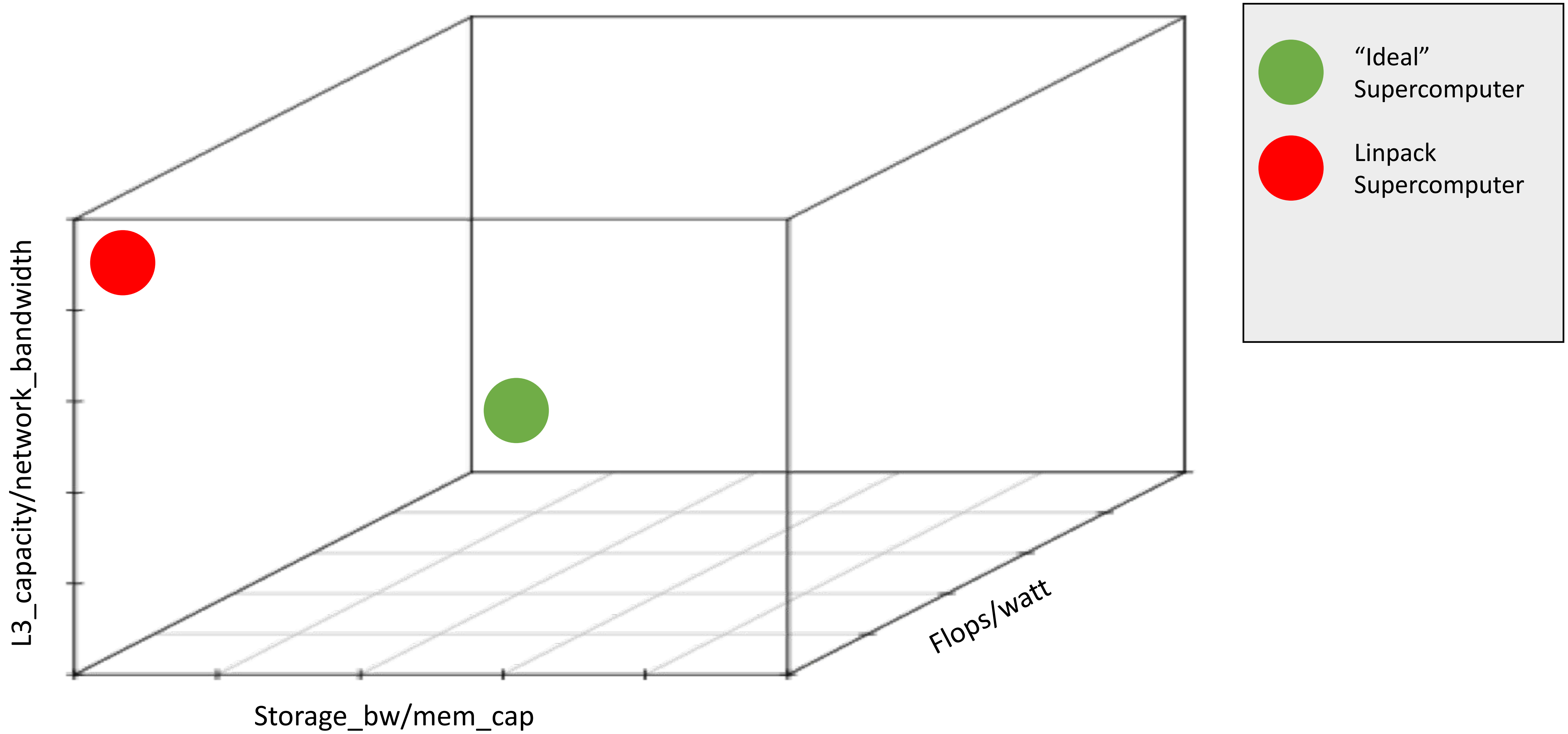
- *Building balanced systems to improve system efficiency and user productivity*

IO500 | A Legitimate Concern About Linpack

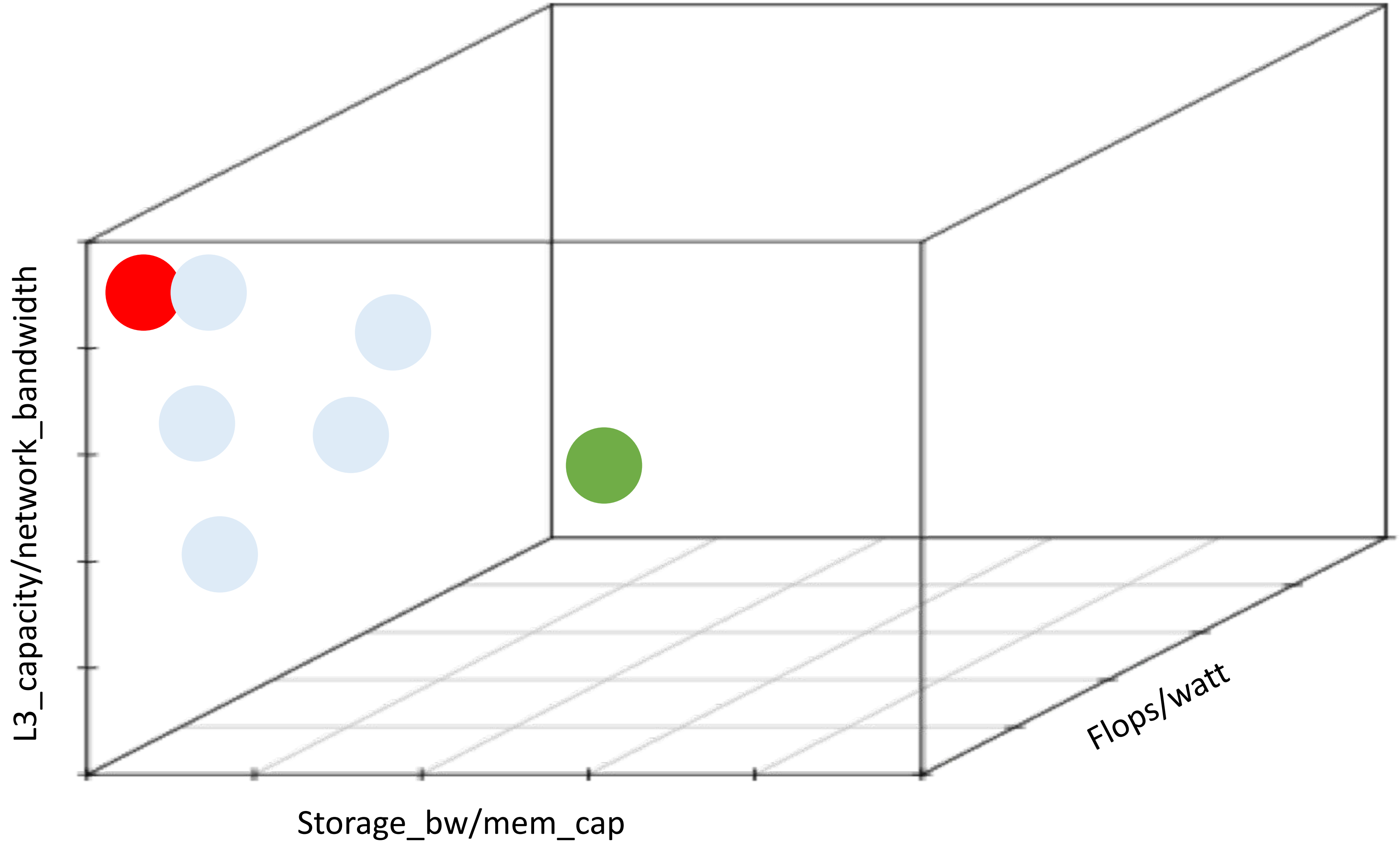


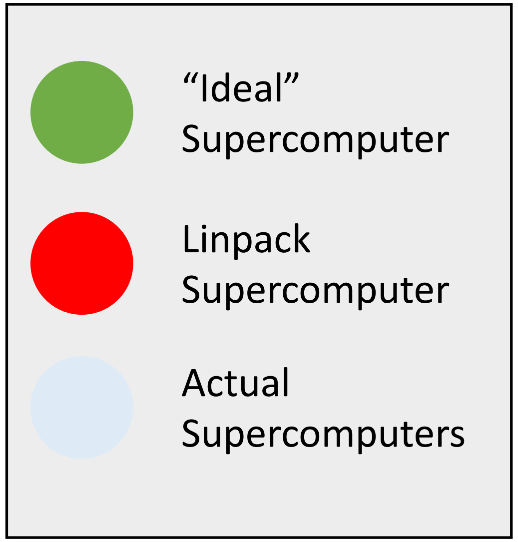
* Please do not interpret axes literally.
Just examples illustrating multi-variable complexity.

IO500 | A Legitimate Concern About Linpack

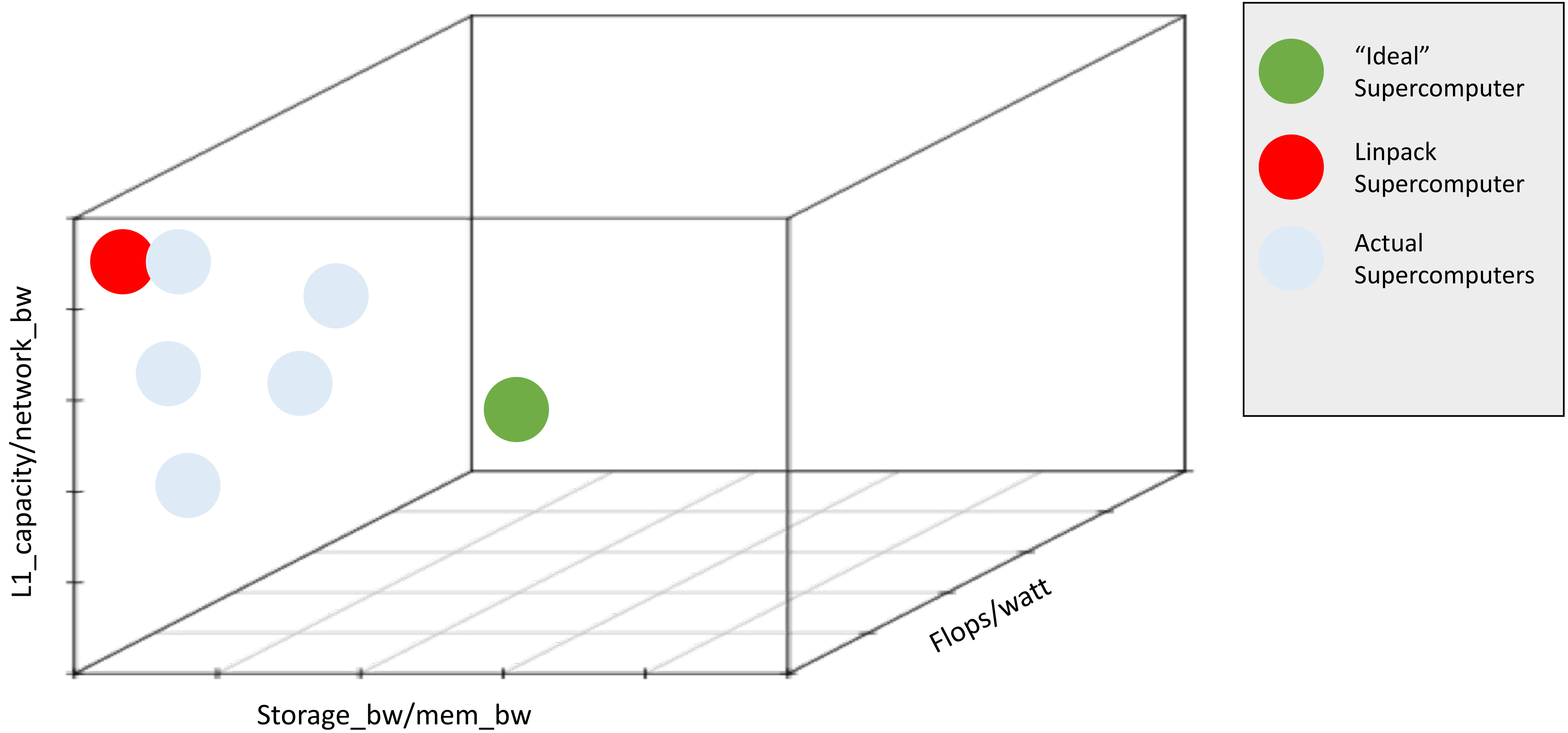


IO500 | A Legitimate Concern About Linpack

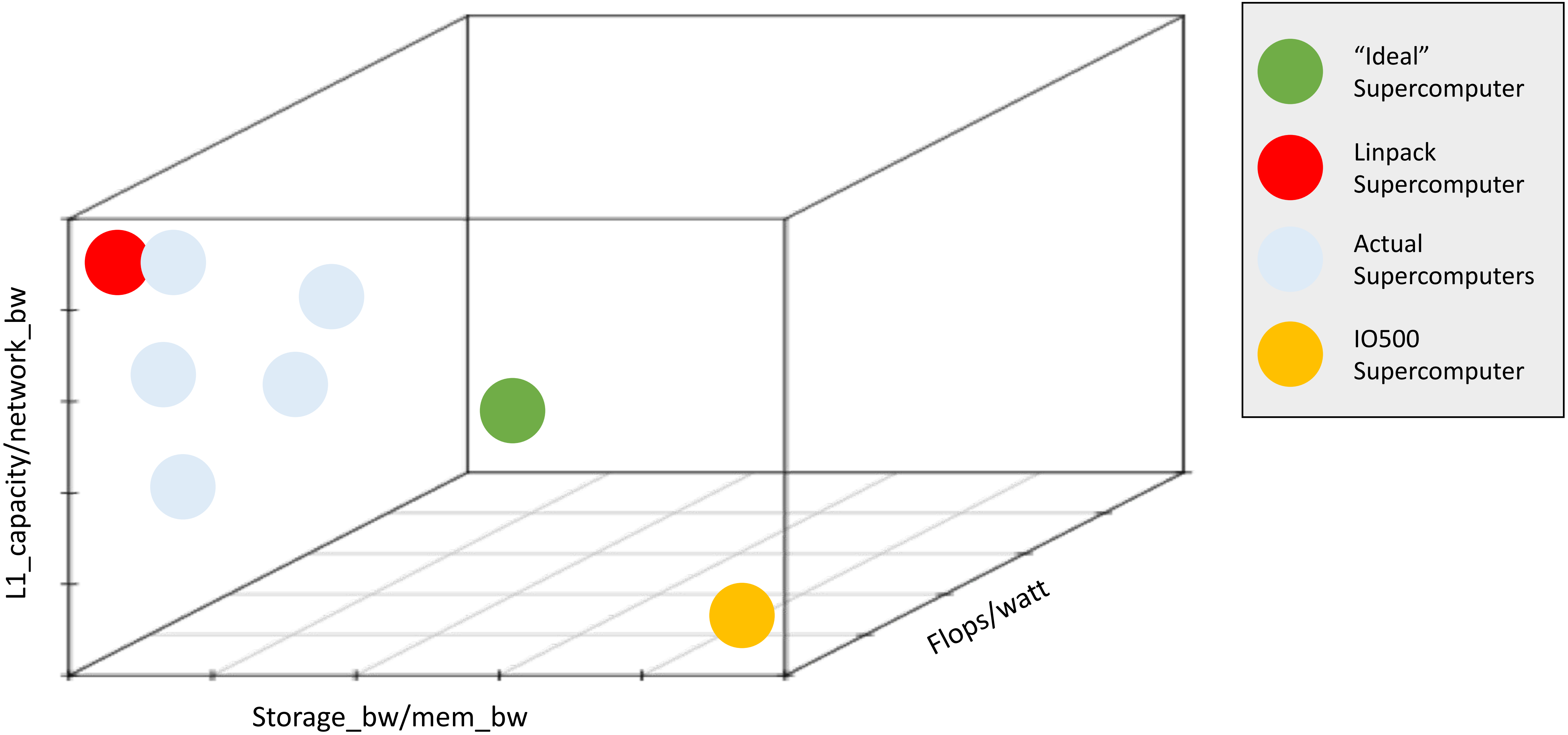




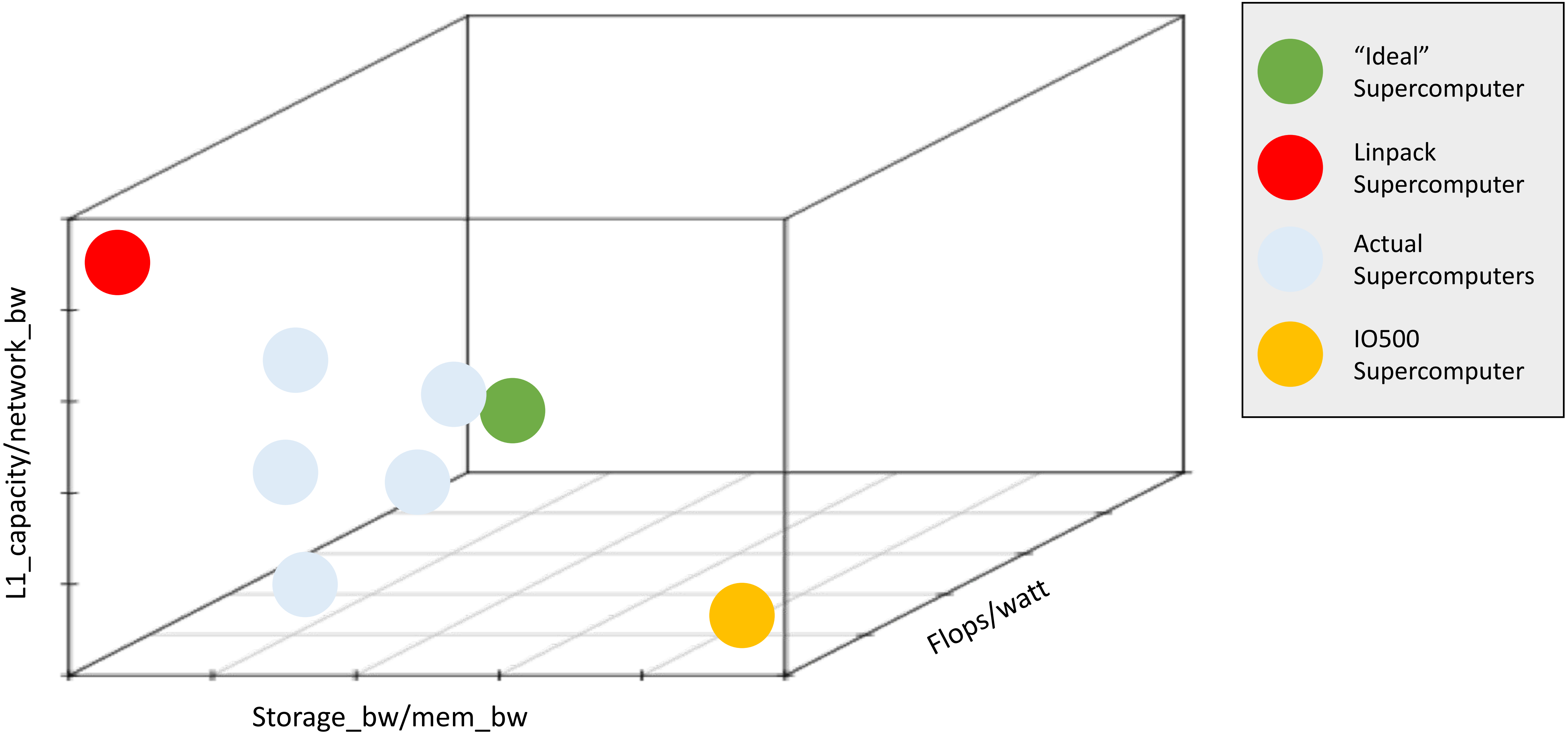
IO500 | IO500 Restores Balance



IO500 | IO500 Restores Balance



IO500 | IO500 Restores Balance



IO500 | IO500 is Balanced

- Hero bandwidth
  - Write and read
- Anti-hero bandwidth
  - Write and read
- Hero metadata
  - Create, stat, delete
- Anti-hero metadata
  - Create, stat, read, delete
- And a namespace search
  - Search

geometric mean → bandwidth score

geometric mean → metadata score

geometric mean → total score

IO500 | Bounding Box of Expectation

- "We tried 20 years ago. Impossible to create a single representative benchmark."
  - Great point! We won't try. Our bounding box includes them all.

Best Possible Metadata Rates

IO500 Bounding Box of Expectation

Worst Possible Metadata Rates

Worst Possible Bandwidth *Best Possible Bandwidth*

<u>BOLD CLAIM</u>

IO500 cannot be gamed.

Whatever you do to improve your IO500 score will result in a better storage system for applications.

Prove me wrong. ☺

IO500 | Current Status of the Benchmark

- Stonewall makes it easier to run than it was previously
  - Importantly captures the straggler effect
- ldiskfs limitation makes mdtest_hard_write difficult
  - Mdtest has been modified to address this
  - [But doesn't yet work with stonewall]
- Parallel rm needed for cleanup
- Several other open feature requests
  - https://github.com/VI4IO/io-500-dev

IO500 | Thanks for all the submissions!

- 54 new submissions from 19 institutions; up to 67 new submissions

```
mysql> select information__institution,count(*) as count from io500 where information__submission_date gte '2018-06';
```

information__institution	count
Joint Institute for Nuclear Research	1
Sandia National Laboratories	1
DKRZ	1
Queen Mary, University Of London	1
Google and DDN	1
EPCC	1
Penguin Computing Advanced Solutions Group	1
DDN	1
QMUL	1
Nemours	2
Google	2
Korea Institute of Science and Technology Information (KISTI)	2
JCAHPC	2
CERN	2
ORNL	2
WekaIO	2
KAUST	2
STFC	3
University of Cambridge	3
Clemson University	23

20 rows in set (0.04 sec)

Maybe enough for some analysis?

All data is available for analysis.

10 Node Challenge

This is the official result list from SC 2018 for the 10 Node Challenge. The list shows all qualifying 10 node results.

IO500

#	information							io500		
	institution	system	storage vendor	filesystem type	client nodes	client total procs	data	score	bw	md
									GiB/s	kIOP/s
1	Oak Ridge National Laboratory	Summit	IBM	Spectrum Scale	10	160	zip	70.63	9.84	506.93
2	WekaIO		WekaIO		10	700	zip	67.79	27.05	169.93
3	DDN	Bancholab	DDN	Lustre	10	240	zip	31.78	6.33	159.41
4	IBM	Sonasad	IBM	Spectrum Scale	10	10	zip	24.24	4.57	128.61
5	KAUST	Shaheen II	Cray	DataWarp	10	80	zip	13.99	14.45	13.53
6	Google and DDN	Lustre on GCP	Google	Lustre	10	80	zip	12.87	4.30	38.52
7	Clemson University	ofsdev	Dell	BeeGFS	10	80	zip	11.58	2.32	57.89
8	Queen Mary; University Of London	Apocrita	E8	GPFS	10	240	zip	9.79	4.32	22.21
9	Clemson University	ofsdev	Dell	Lustre	10	40	zip	8.18	1.90	35.26
10	Clemson University	Palmetto	Dell	BeeGFS	10	10	zip	7.51	2.32	24.34
11	DKRZ	Mistral	Seagate	Lustre	10	80	zip	5.35	1.05	27.33
12	Queen Mary, University Of London	Apocrita	DDN	GPFS	10	240	zip	3.73	0.87	15.94
13	EPCC	Archer	Seagate	Lustre	10	80	zip	3.70	0.77	17.84
14	Sandia National Laboratories	Ghost	IBM	GPFS	10	100	zip	0.32	0.05	2.00

2018-11

This is the official list from SC 2018. The list shows the best result for a given combination of system/institution/filesystem.

#	information							io500		
	institution	system	storage vendor	filesystem type	client nodes	client total procs	data	score	bw GiB/s	md kIOP/s
1	Oak Ridge National Laboratory	Summit	IBM	Spectrum Scale	504	1008	zip	366.47	88.20	1522.69
2	Korea Institute of Science and Technology Information (KISTI)	NURION	DDN	IME	2048	4096	zip	160.67	554.23	46.58
3	University of Cambridge	Cumulus	Dell EMC	DAC-Lustre	184	2944	zip	158.71	71.40	352.75
4	JCAHPC	Oakforest-PACS	DDN	IME	2048	16384	zip	137.78	560.10	33.89
5	KAUST	ShaheenII	Cray	DataWarp	1024	8192	zip	77.37	496.81	12.05
6	University of Cambridge	Data Accelerator	Dell EMC	DAC-BeeGFS	184	5888	zip	74.58	58.81	94.57
7	Google	Exascaler on GCP	Google	Lustre	120	960	zip	56.77	23.06	139.74
8	KAUST	ShaheenII	Cray	Lustre	1000	16000		41.00*	54.17	31.03*
9	JSC	JURON	ThinkparQ	BeeGFS	8	64		35.77*	14.24	89.81*
10	DKRZ	Mistral	Seagate	Lustre	100	1000		32.15	22.77	45.39

Radar Chart

This is the official list from SC 2018 with a radar chart and controls to manipulate the ranking. The list shows all results.

#	information								io500		
	submission date	institution	system	storage vendor	filesystem type	client nodes	client total procs	data	score	bw GiB/s	md kIOP/s
1	2018-11-09	Oak Ridge National Laboratory	Summit	IBM	Spectrum Scale	504	1008	zip	366.47	88.20	1522.69
2	2018-11-02	Korea Institute of Science and Technology Information (KISTI)	NURION	DDN	IME	2048	4096	zip	160.67	554.23	46.58
3	2018-11-12	University of Cambridge	Cumulus	Dell EMC	DAC-Lustre	184	2944	zip	158.71	71.40	352.75
4	2018-05-09	JCAHPC	Oakforest-PACS	DDN	IME	2048	16384	zip	137.78	560.10	33.89
5	2017-11	JCAHPC	Oakforest-PACS	DDN	IME	2048	16384	zip	101.48	471.25	21.85
6	2018-11-13	WekaIO	WekaIO	WekaIO		17	935	zip	92.95	37.39	231.05
7	2018-06-21	KAUST	ShaheenII	Cray	DataWarp	1024	8192	zip	77.37	496.81	12.05
8	2018-10-25	University of Cambridge	Data Accelerator	Dell EMC	DAC-BeeGFS	184	5888	zip	74.58	58.81	94.57
9	2017-11	KAUST	ShaheenII	Cray	DataWarp	300	2400		70.90*	151.53	33.17*
10	2018-11-09	Oak Ridge National Laboratory	Summit	IBM	Spectrum Scale	10	160	zip	70.63	9.84	506.93

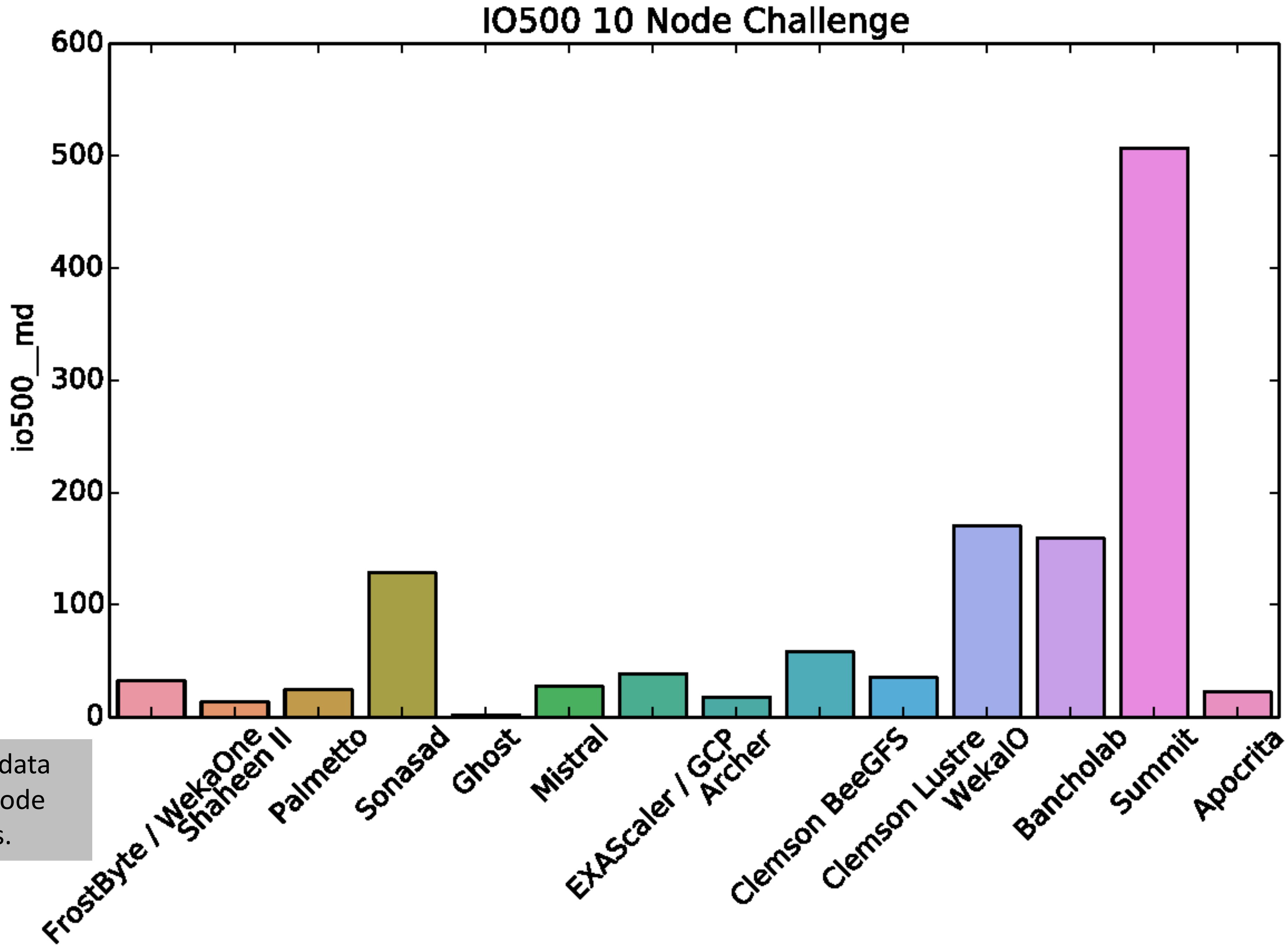
Preliminary Analyses

- Now that we have lots of data, the following slides will attempt some preliminary analyses and suggest different ways of using IO500 as well
- There are the following sections
  - Analysis of the 10 Node Challenge
  - Analysis of the Overall Top Five Systems
  - Analysis of the Cambridge apples-apples results comparing untuned BeeGFS, untuned Lustre, and tuned Lustre – highlights also the value of Lustre DNE2
  - Analysis of the Exascaler on Google Cloud Platform results showing what happens with different numbers of clients, metadata servers, and object servers
  - Analysis of the straggler effect and whether some filesystems might be less sensitive than others – no spoilers!
  - Analysis of degradation from easy to hard and whether some filesystems might be less sensitive than others – no spoilers!
  - Analysis of the JCAHPC results showing the impact of upgrading their IME version
  - Analysis of the Nemour results showing the value of using IO500 for regression testing
  - Analysis of the IME results further showing the impact of the IME version (maybe)
  - A huge set of extra bonus graphs with no analysis – an exercise for the reader!
- Caveats!
  - There is probably not enough data to be statistically significant. I might imagine trends that don't exist.
  - Broad claims suggesting filesystem comparisons are probably not valid.
  - To any offended file system developers out there, I apologize. Please correct any mistakes and explain any confusion!

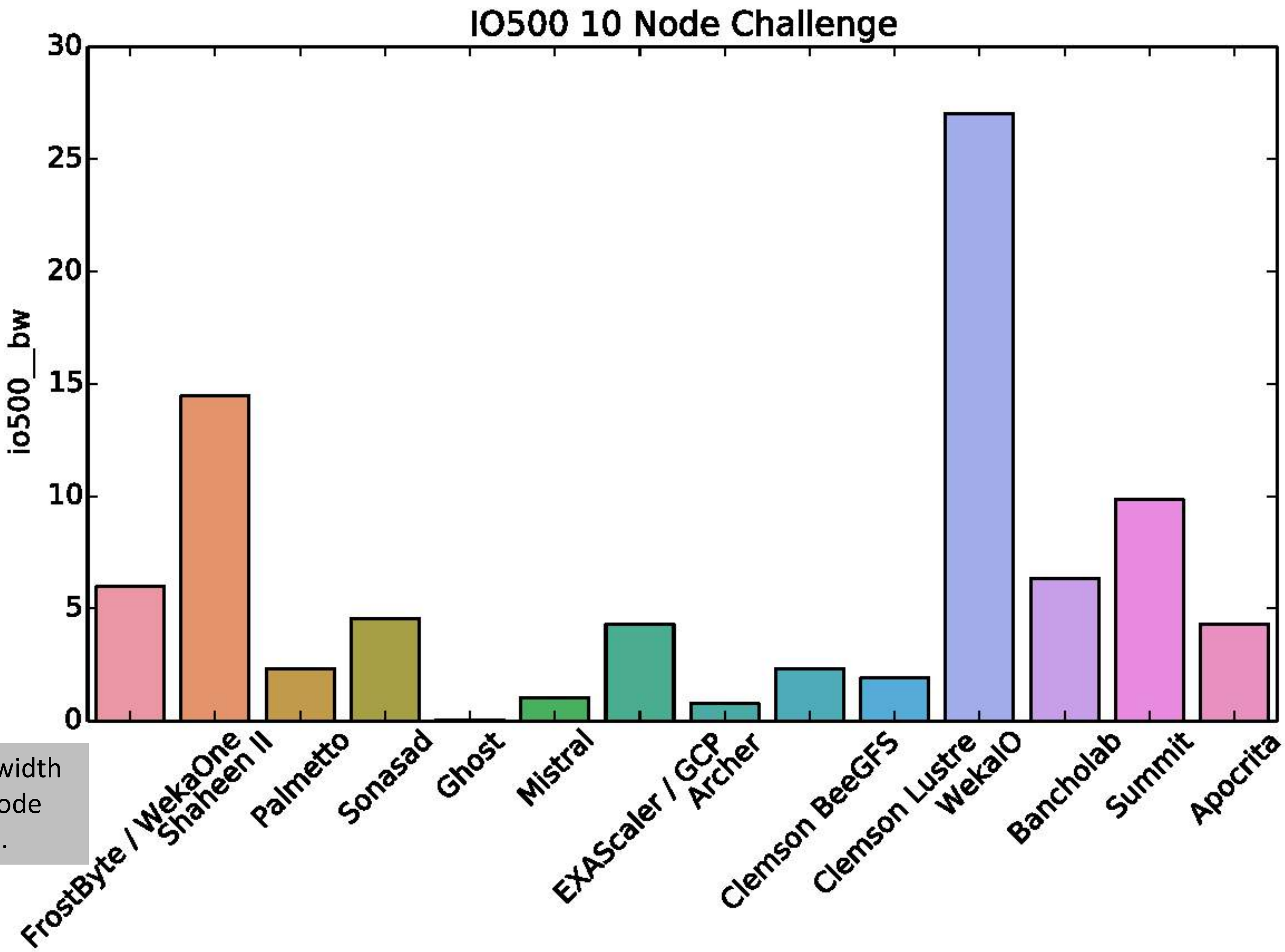
Ten Node Challenge

- We introduced a “Ten Node Challenge” this year in an attempt to encourage small systems to submit
- It was successful; we had 14 submissions!

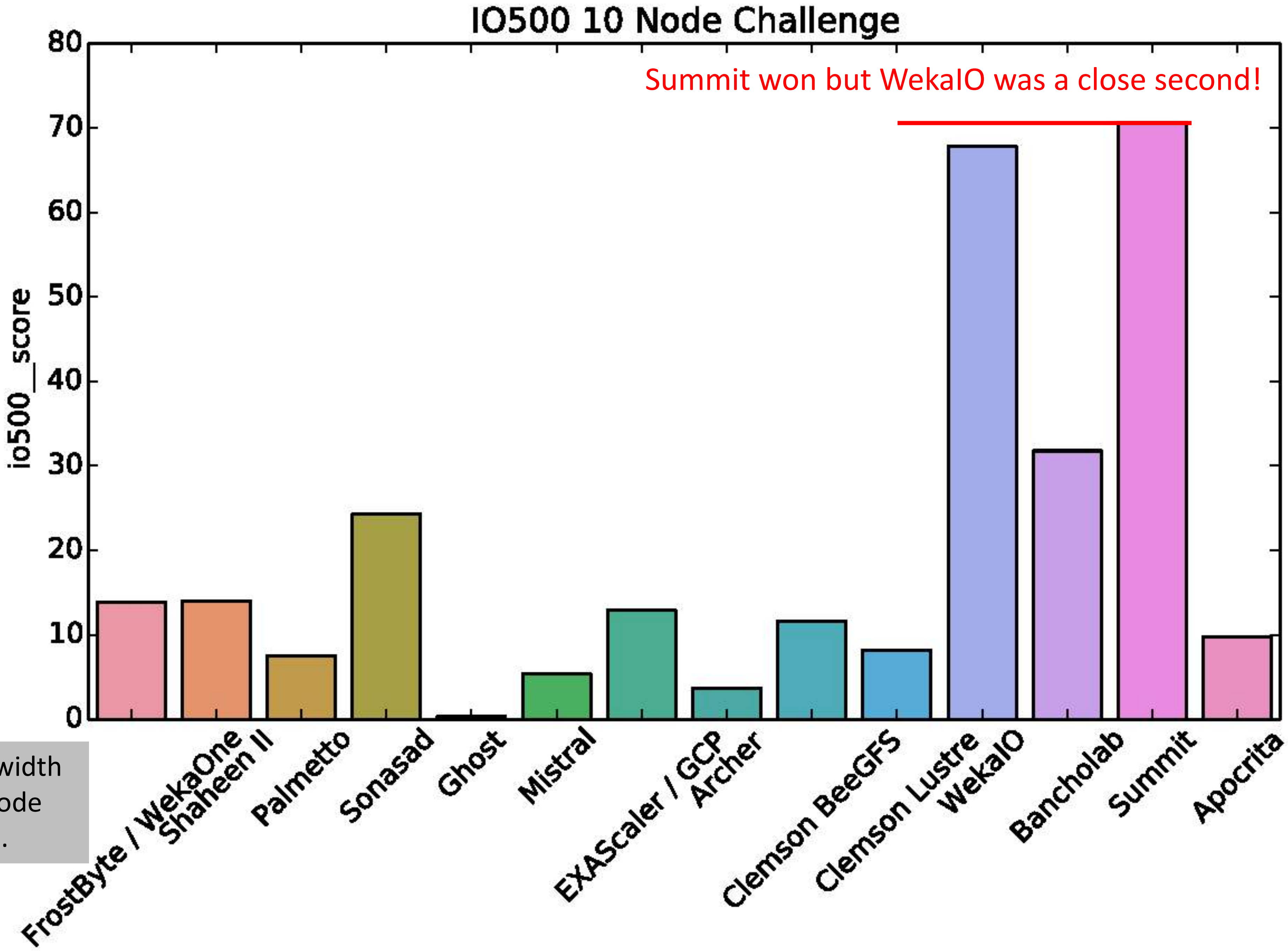
```
[mysql> select information__submission_date,information__system from io500 where information__client_nodes=10 order by information__submission_date;
+-----------------------------+----------------------+
| information__submission_date | information__system |
+-----------------------------+----------------------+
| 2017-11-01                  | Sonasad              |
| 2018-08-22                  | Shaheen II           |
| 2018-11-01                  | Palmetto             |
| 2018-11-03                  | Mistral              |
| 2018-11-05                  | Clemson BeeGFS       |
| 2018-11-07                  | Apocrita             |
| 2018-11-08                  | Clemson Lustre       |
| 2018-11-08                  | Bancholab            |
| 2018-11-09                  | Apocrita             |
| 2018-11-09                  | Archer               |
| 2018-11-09                  | Summit               |
| 2018-11-10                  | Ghost                |
| 2018-11-11                  | FrostByte            |
| 2018-11-11                  | EXAScaler / GCP      |
| 2018-11-14                  | WekaIO               |
+-----------------------------+----------------------+
15 rows in set (0.08 sec)
```



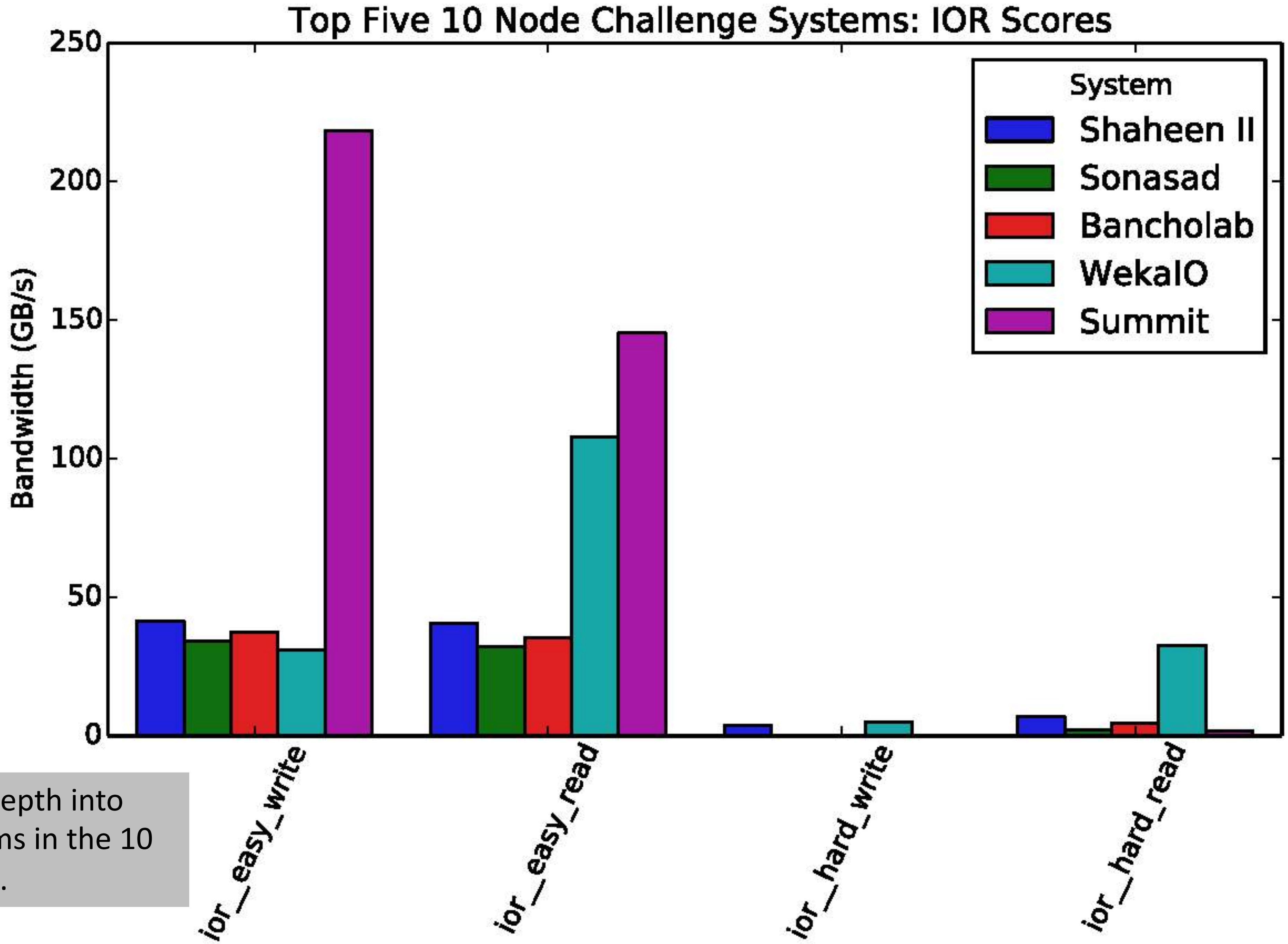
The overall metadata score for all 10 Node Challenge entries.



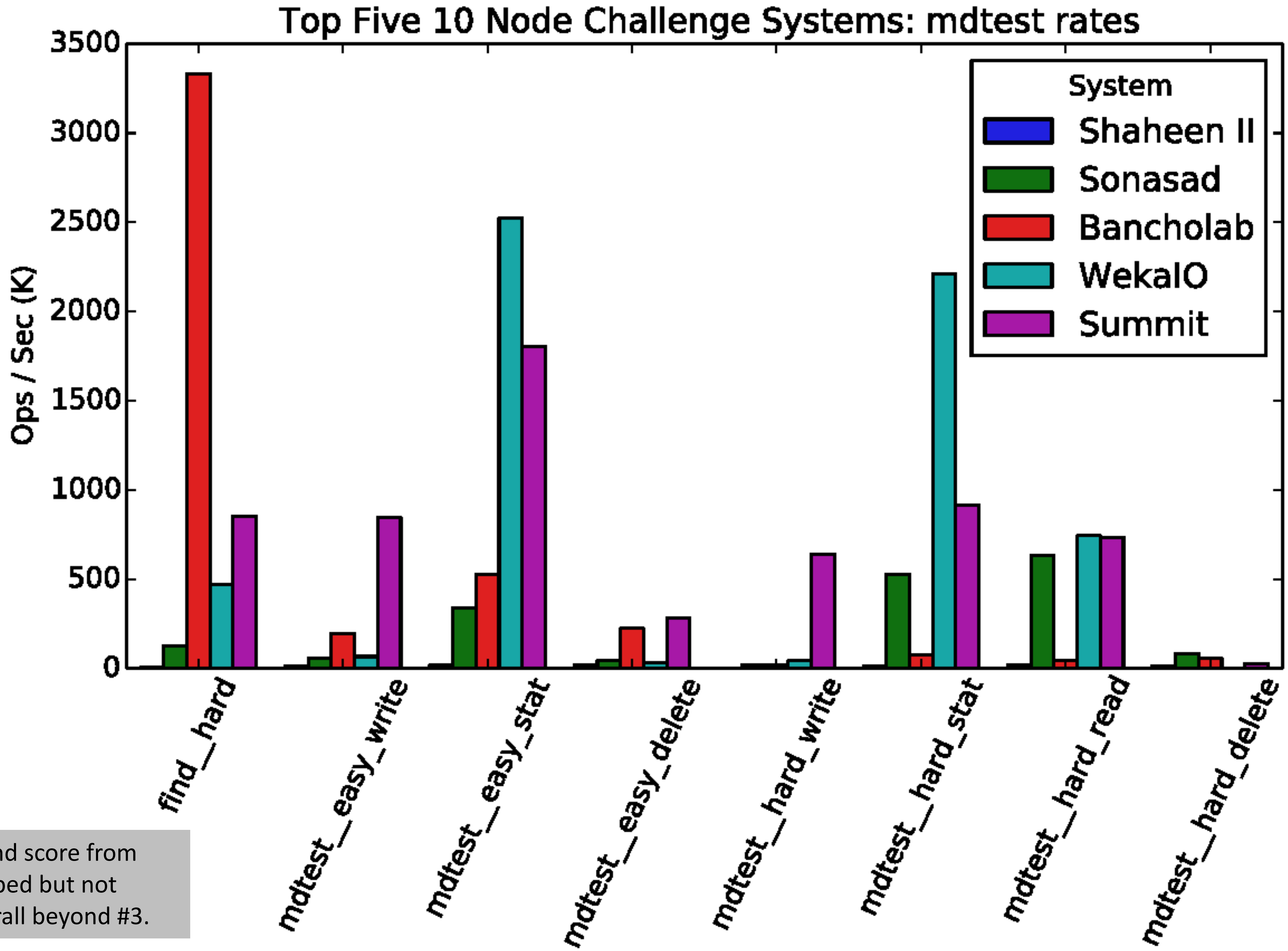
The overall bandwidth score for all 10 Node Challenge entries.



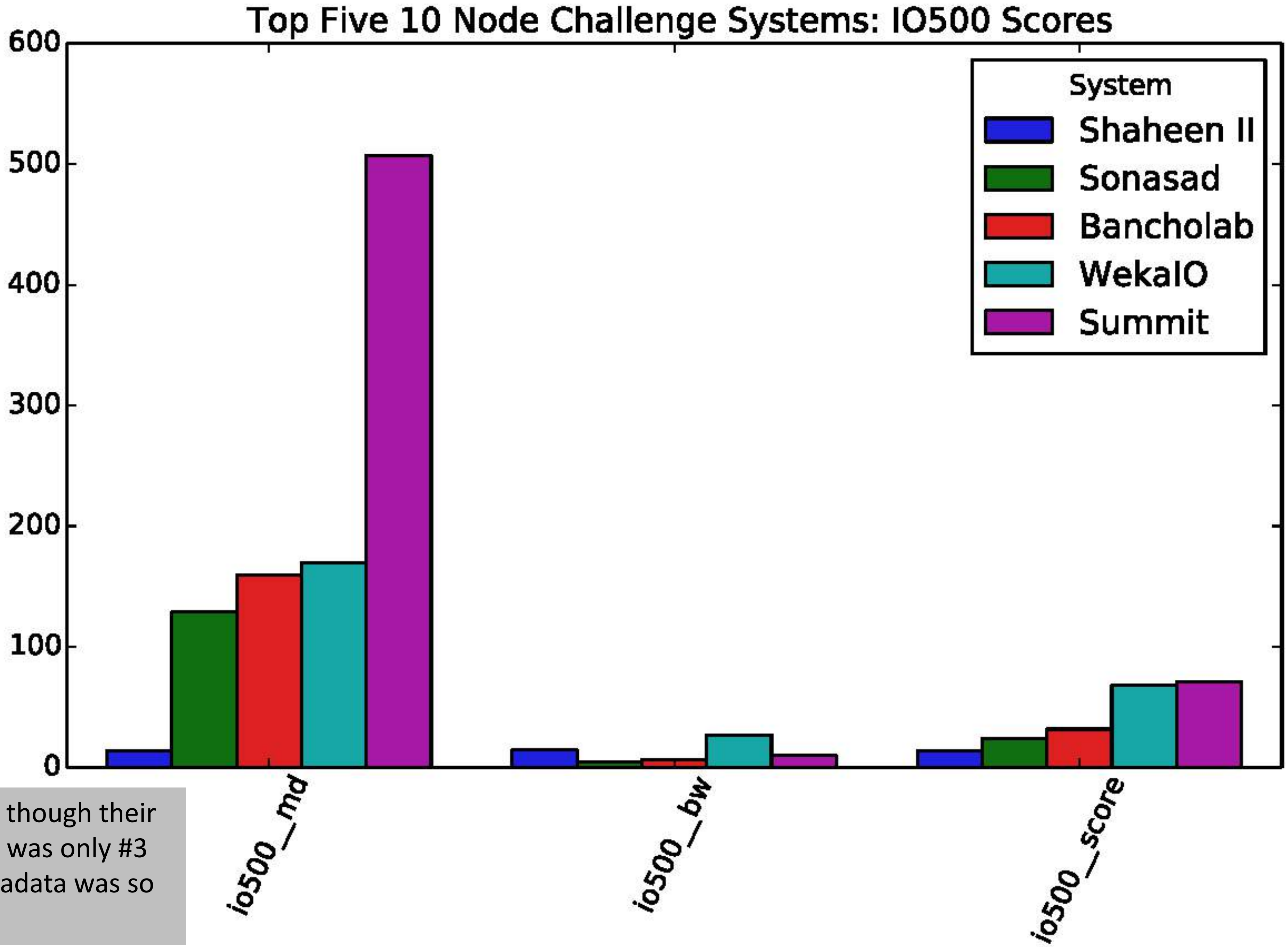
The overall bandwidth score for all 10 Node Challenge entries.



Here we go in-depth into the top-5 systems in the 10 Node Challenge.



A very impressive find score from the DDN Lustre testbed but not enough to lift it overall beyond #3.



Summit won even though their overall bandwidth was only #3 because their metadata was so much better.

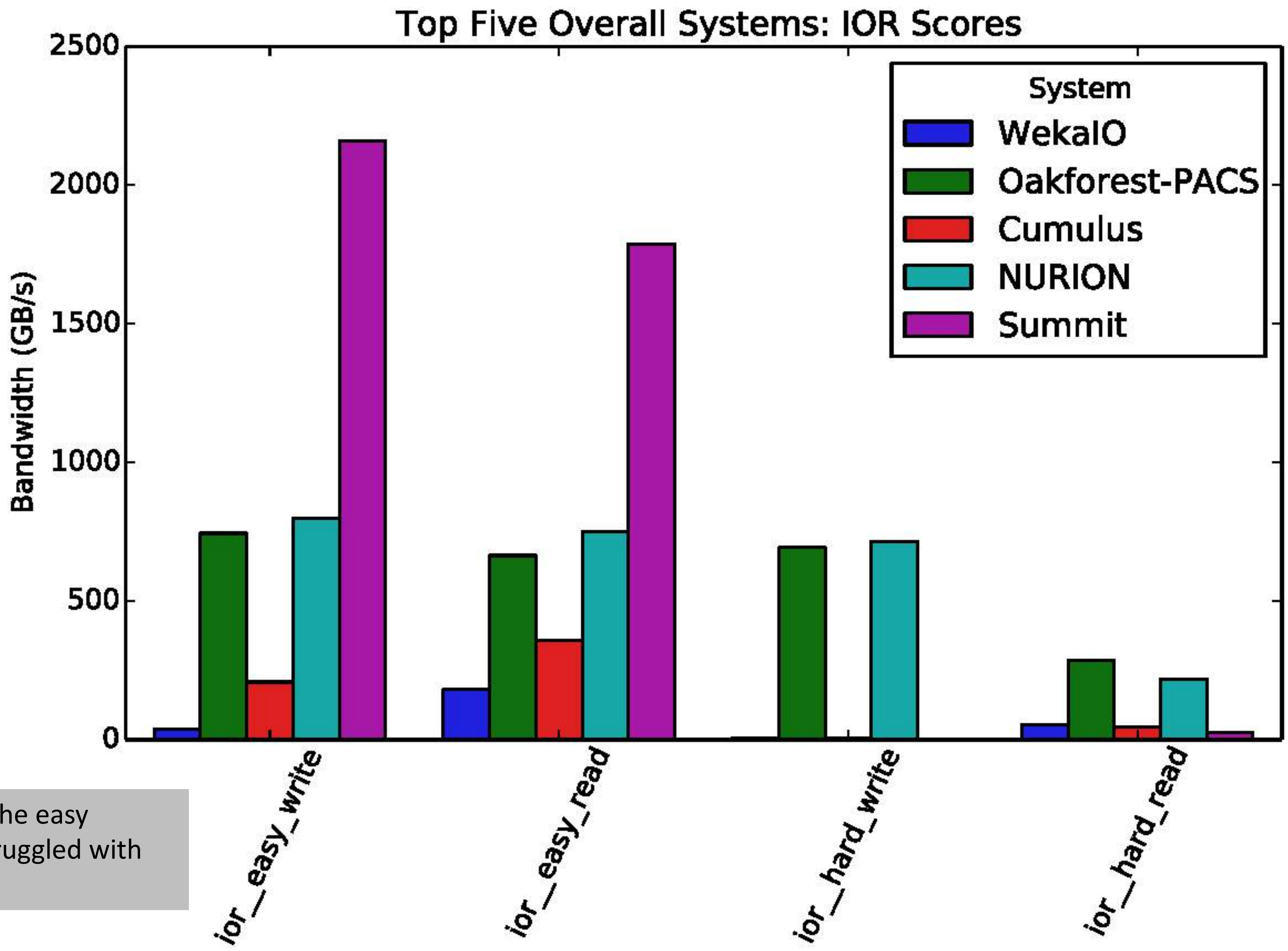
Top Five Overall Systems

- Here we go in-depth into the top five overall systems on the list

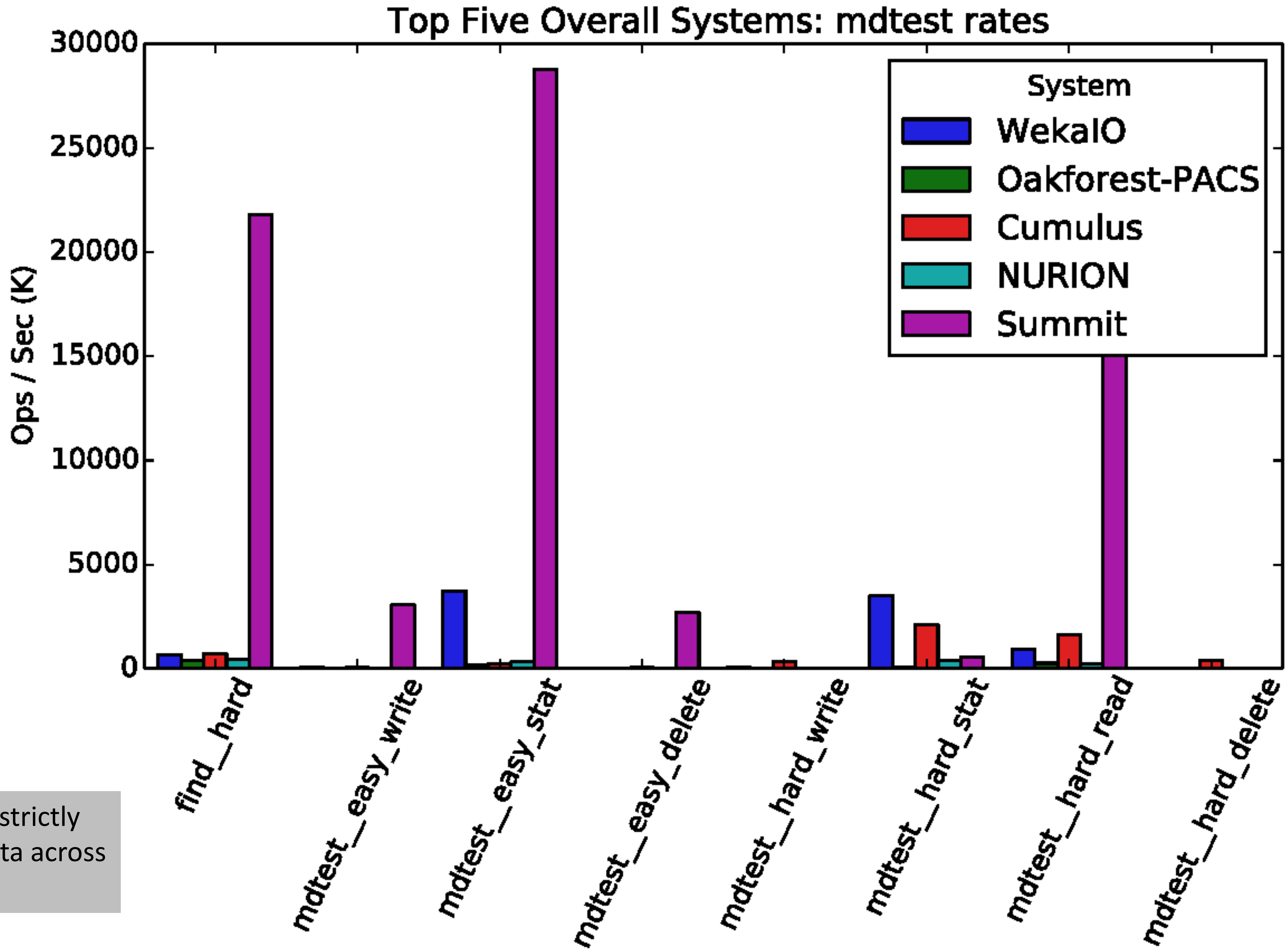
IO-500

This is the official **ranked list**[1)] from SC 2018. The list shows the best result for a given combination of system/institution/filesystem.

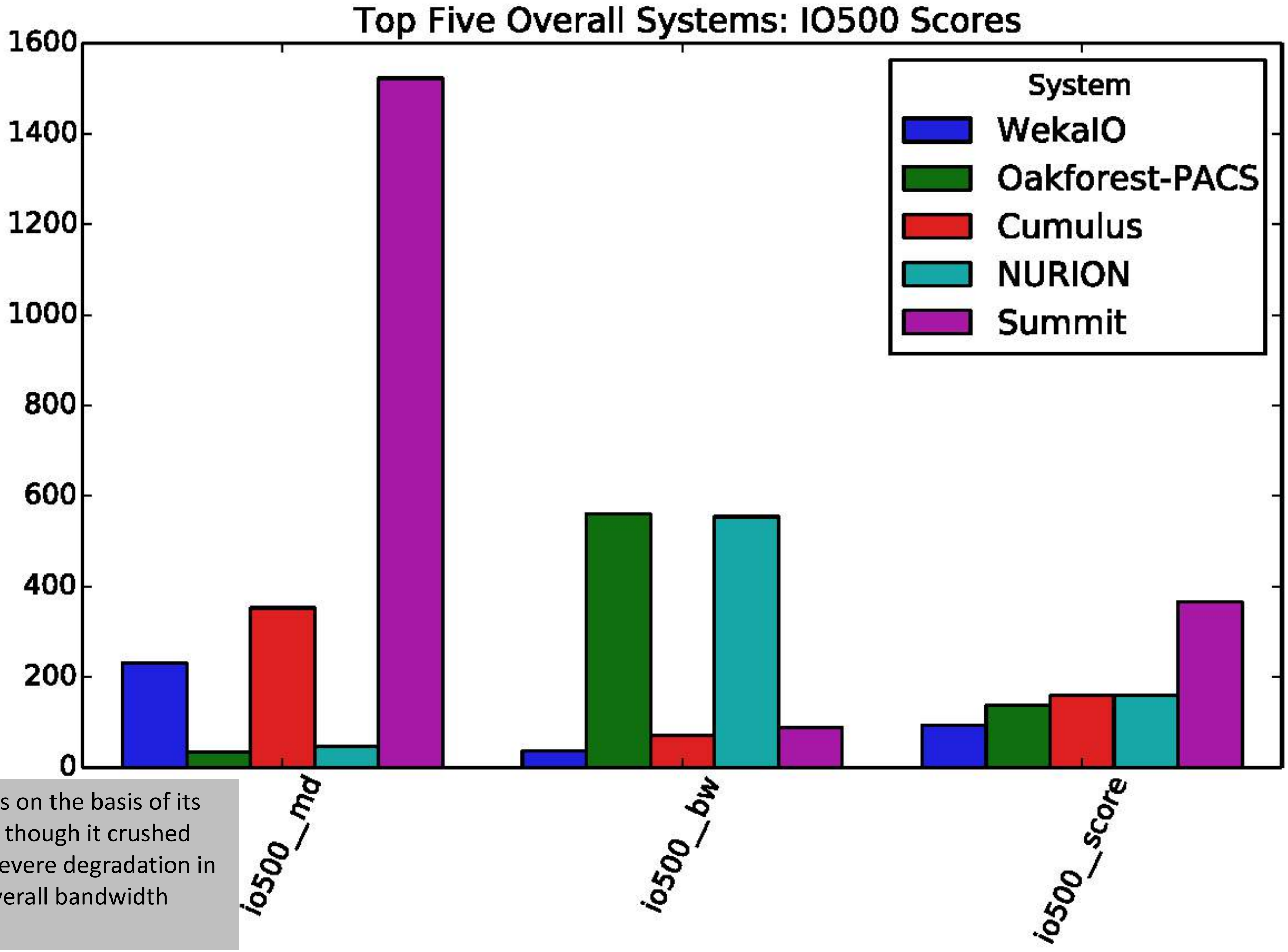
#	information							io500		
	institution	system	storage vendor	filesystem type	client nodes	client total procs	data	score	bw GiB/s	md kIOP/s
1	Oak Ridge National Laboratory	Summit	IBM	Spectrum Scale	504	1008	zip	366.47	88.20	1522.69
2	Korea Institute of Science and Technology Information (KISTI)	NURION	DDN	IME	2048	4096	zip	160.67	554.23	46.58
3	University of Cambridge	Data Accelerator	Dell EMC	Lustre	528	4224	zip	158.71	71.40	352.75
4	JCAHPC	Oakforest-PACS	DDN	IME	2048	16384	zip	137.78	560.10	33.89
5	WekaIO	WekaIO	WekaIO		17	935	zip	92.95	37.39	231.05



Summit crushed the easy bandwidth but struggled with hard.



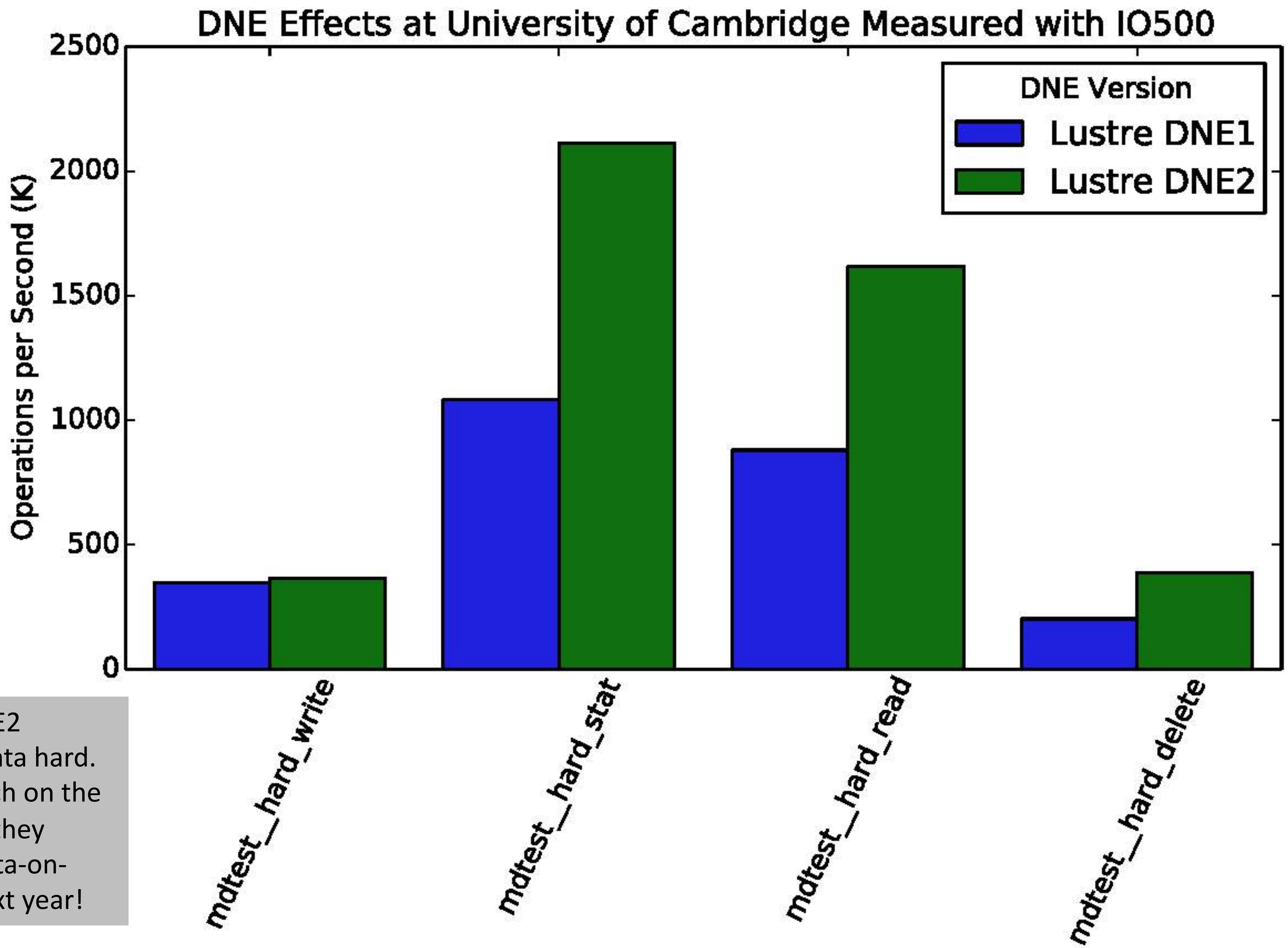
Summit is almost strictly better for metadata across the board.



Yet again, Summit wins on the basis of its metadata score. Even though it crushed easy bandwidth, the severe degradation in hard really hurt the overall bandwidth score.

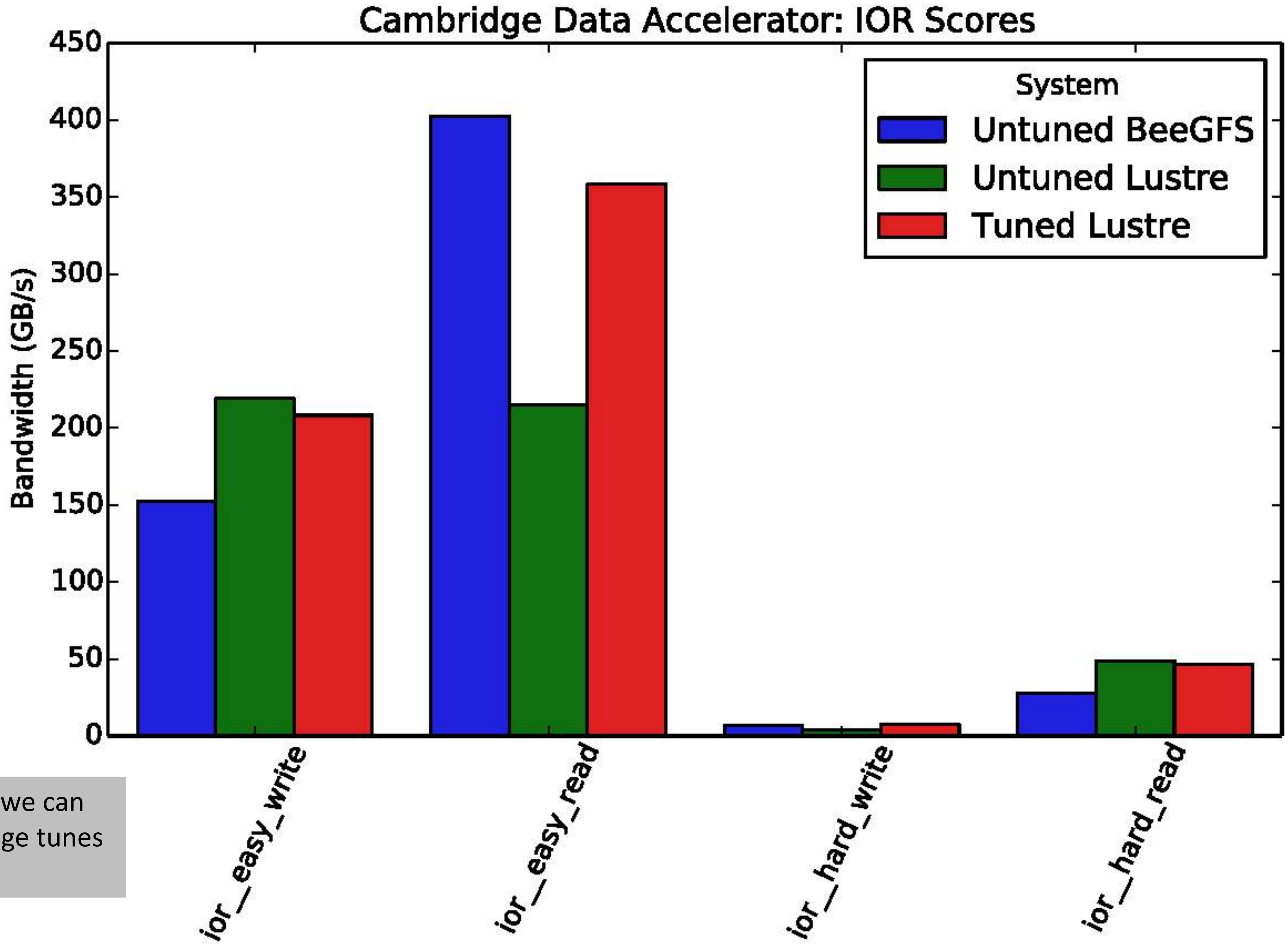
Some Analysis

- University of Cambridge Data Accelerator submitted three results
- We can look at the value of tuning and the impact of Lustre DNE2 on mdtest_hard_*
- Fantastic apples-apples comparison on the exact same hardware

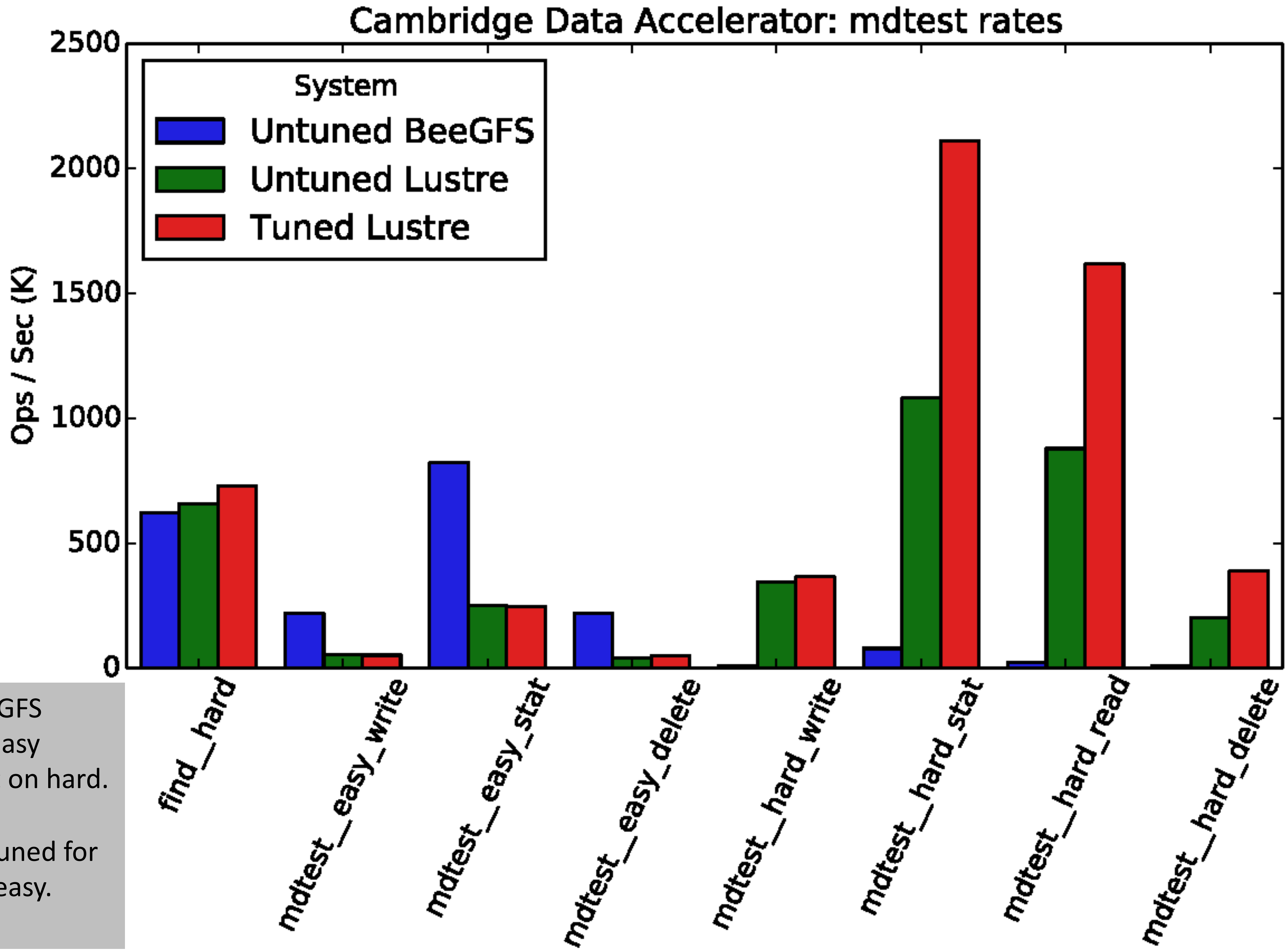


IO500

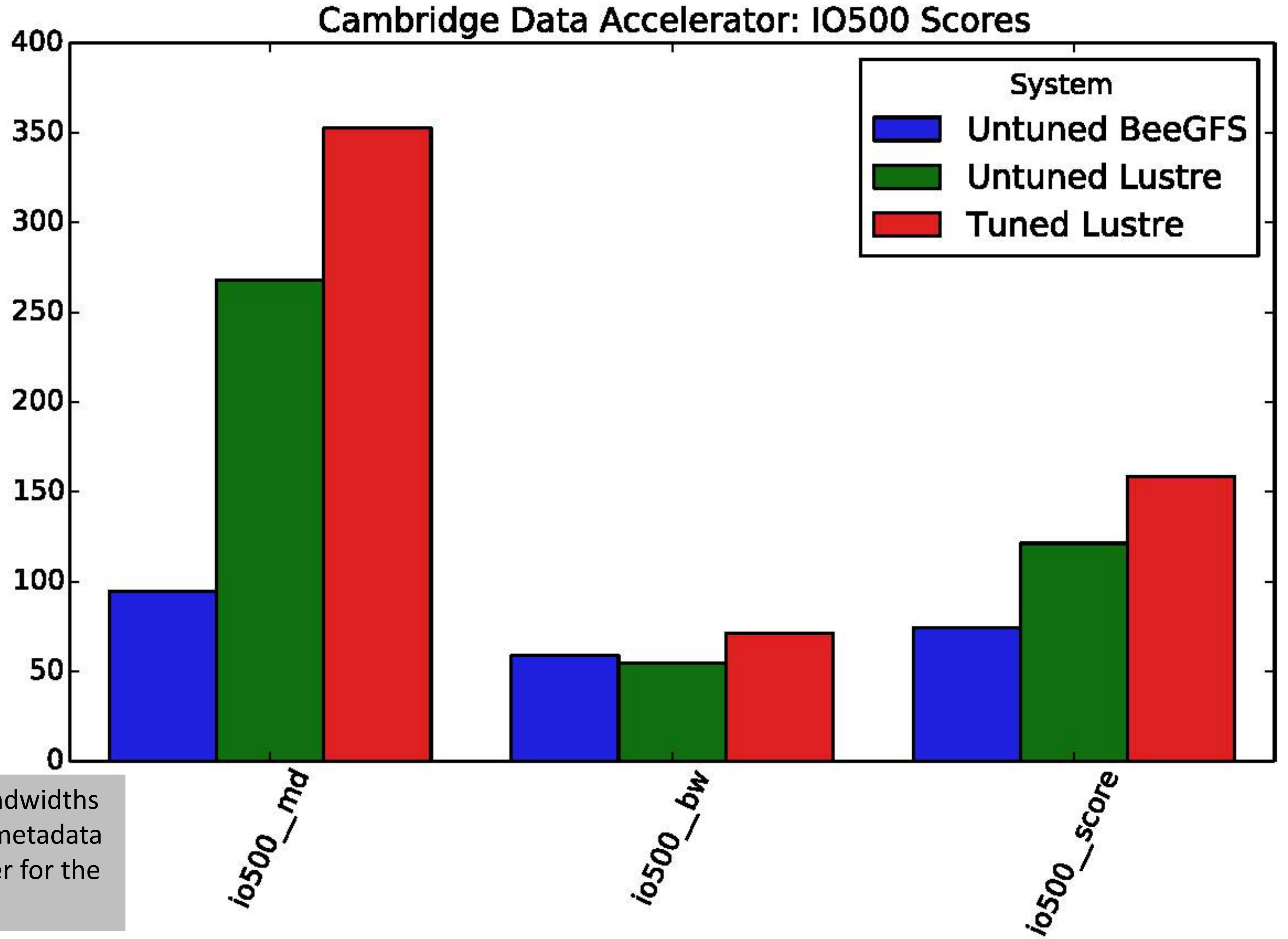
As expected, DNE2 improves metadata hard. But not very much on the write. Probably they weren't using data-on-metadata . . . Next year!



Maybe next year we can see how Cambridge tunes BeeGFS!



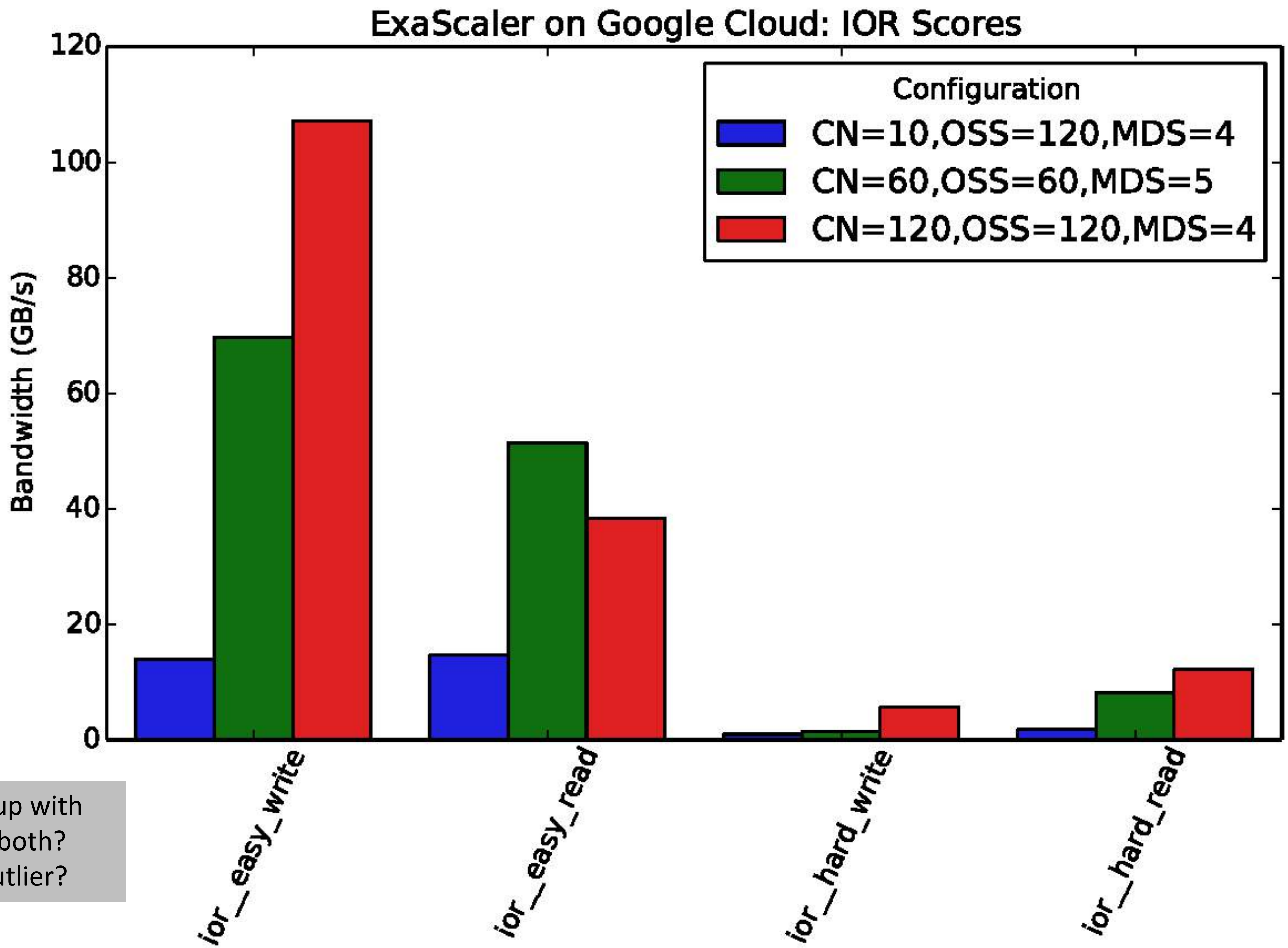
The untuned BeeGFS outperforms on easy metadata but not on hard. However, I think Cambridge only tuned for hard and not for easy. Next year!



Pretty similar bandwidths so yet again the metadata is the prime driver for the overall score.

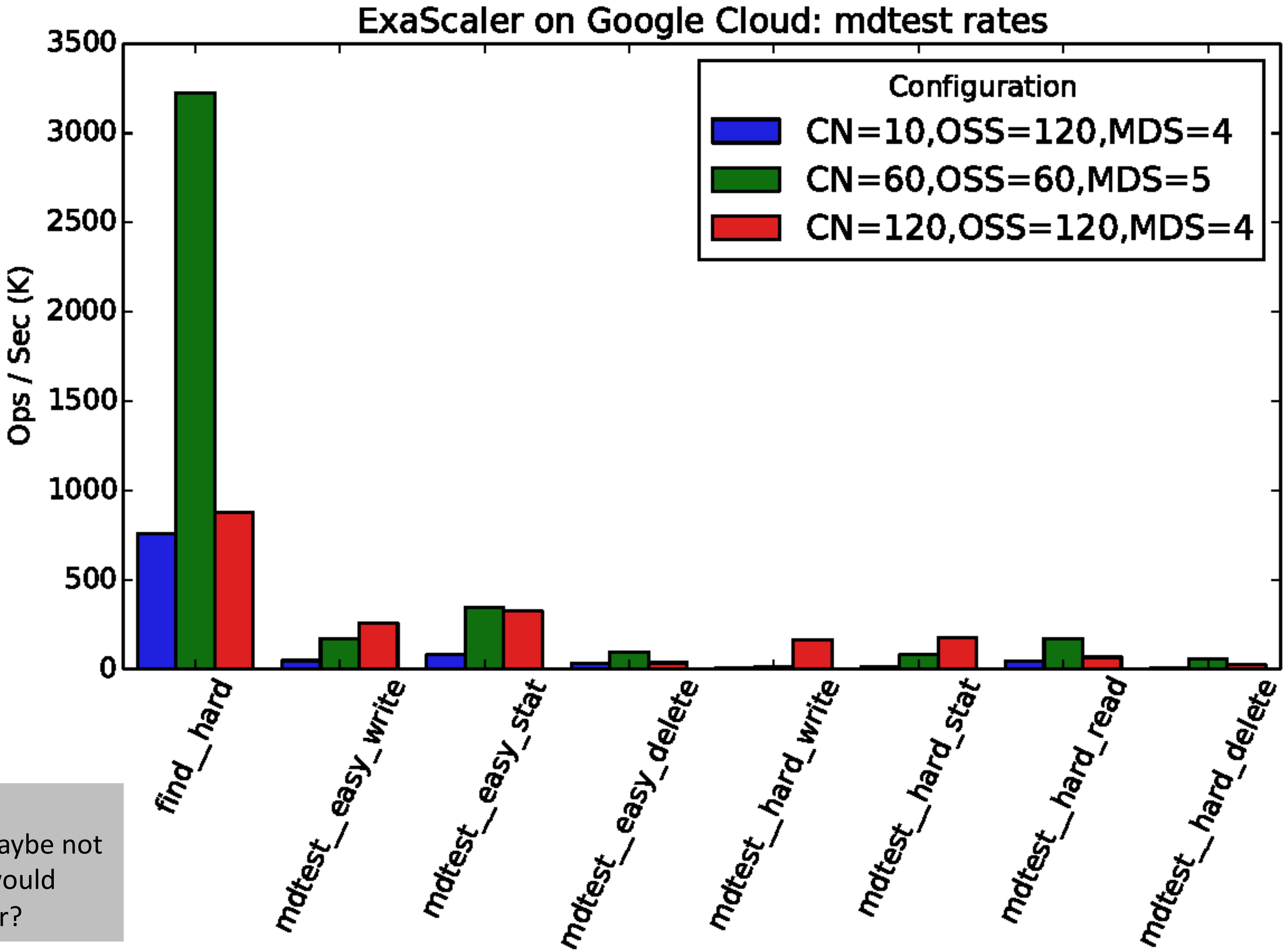
DDN and Google Partnered on Exascaler on GCP

- Three runs allow us to see the varying affects of changing client count, MDS count, and OSS count in the Lustre file system

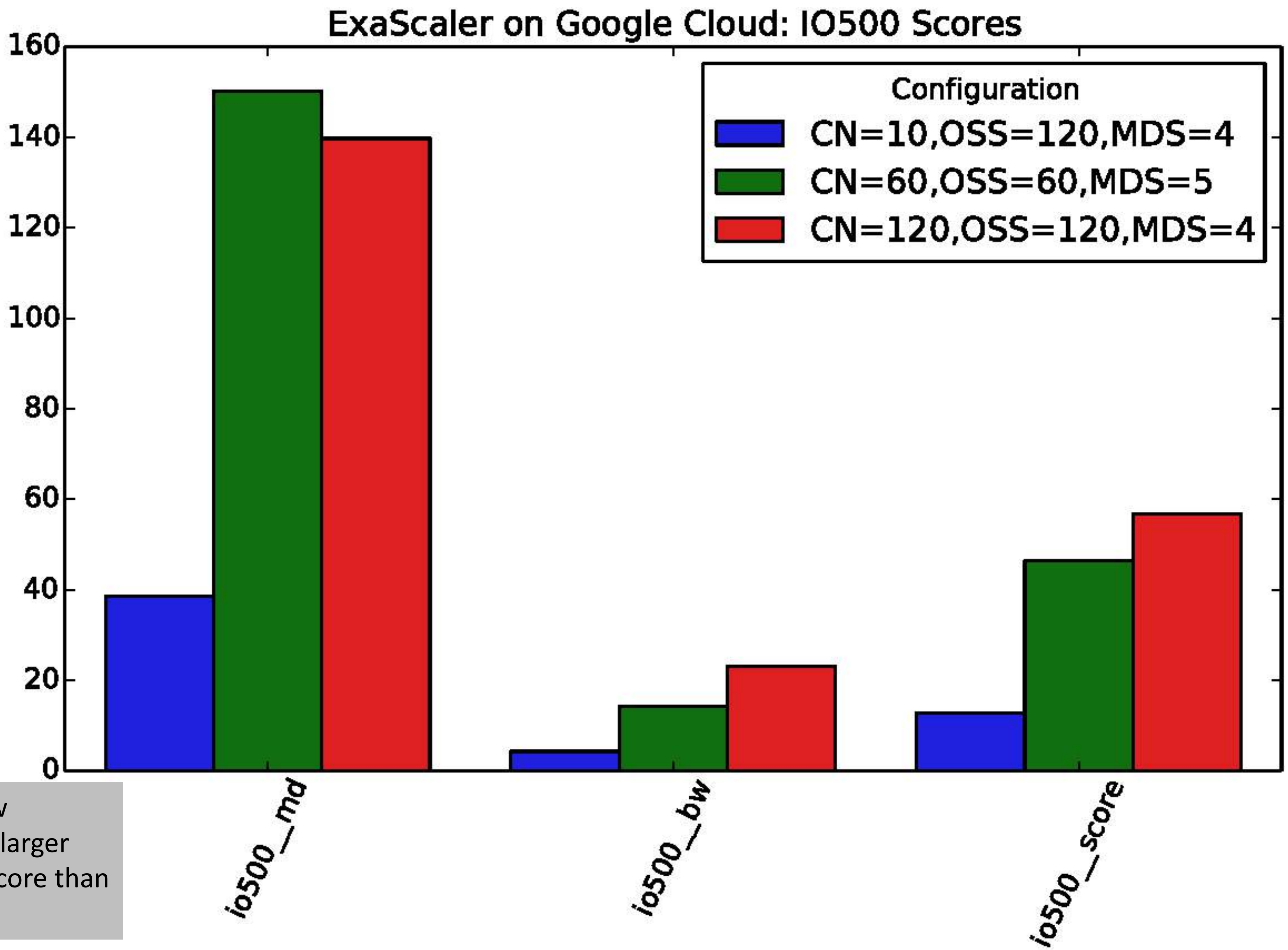


IO500

Bandwidth goes up with clients or OSS or both?
Easy read is an outlier?



More MDS helps metadata. But maybe not as much as find would suggest . . . outlier?



A result where bw seemed to play a larger factor in overall score than metadata

Straggler Analysis

- Reminder of how stonewall works in IO500:

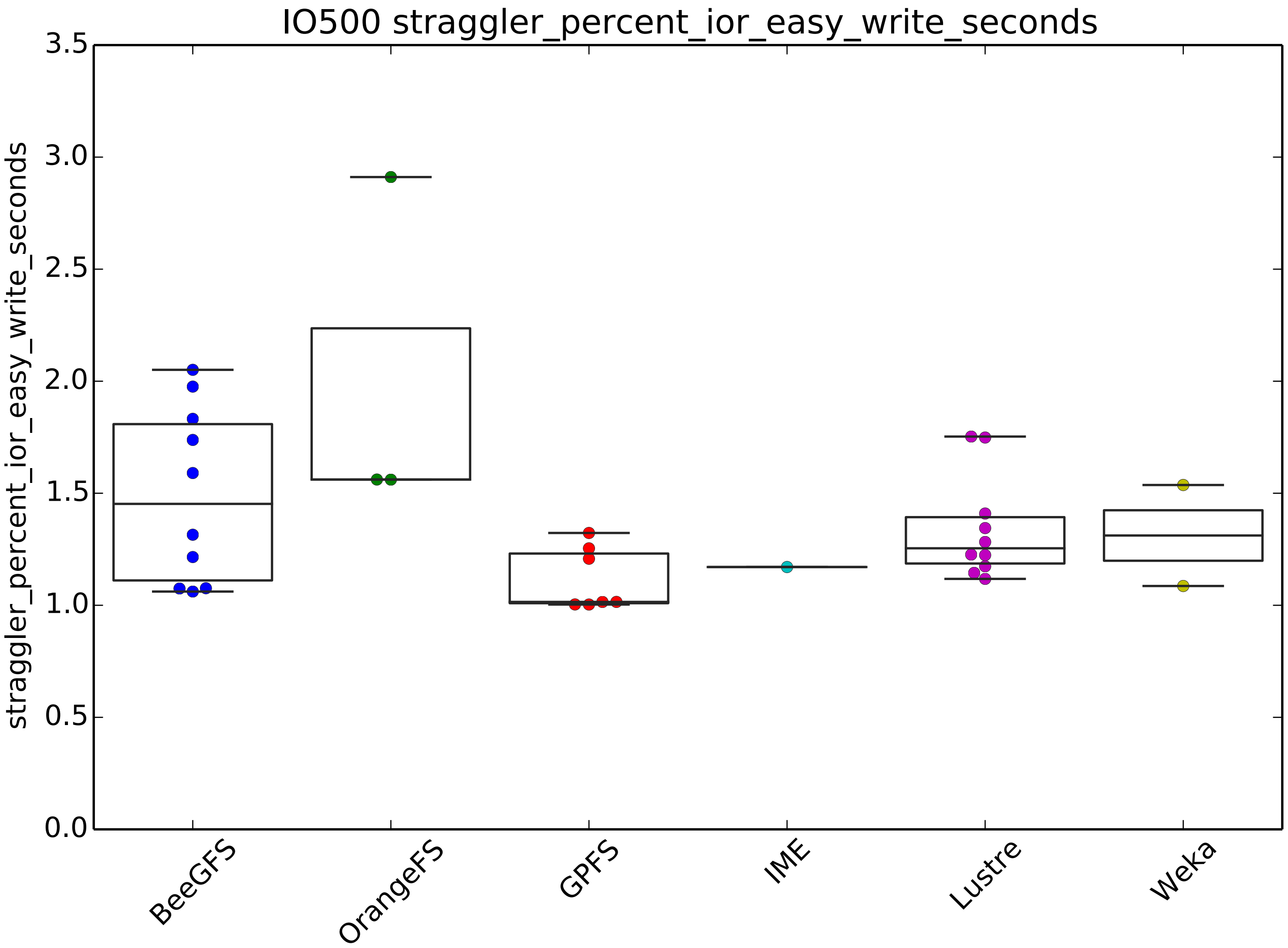
- This is because real codes do fixed amounts of work not fixed amounts of time
- But stonewall is useful for benchmarking!
- We do the wearout to get a realistic measurement in a bounded amount of time

- This then effectively measures how balanced a system is.
- In a perfectly balanced system, everyone will do the same amount of work in the same amount of time and straggler_effect will be 1
- High straggler_effects might indicate imbalanced systems

```
stonewall = 300
myunits = 0
timer = now()
while(true)
   do_work()                          Stonewall phase
   myunits++
   break if now()-timer >= stonewall
maxunits=MPI_Reduce(MAX,myunits,COMM_WORLD)
while myunits < maxunits
    do_work                           Wearout phase
    myunits++
elapsed = now()-timer
straggler_effect = elapsed / stonewall
```

Using pandas, sql, matplotlib, and seaborn python modules, all results using stonewall are grouped by filesystem and there is one point for each result. The boxes show median, quartiles, and max-min.

GPFS seems to have the least imbalance for ior easy write.



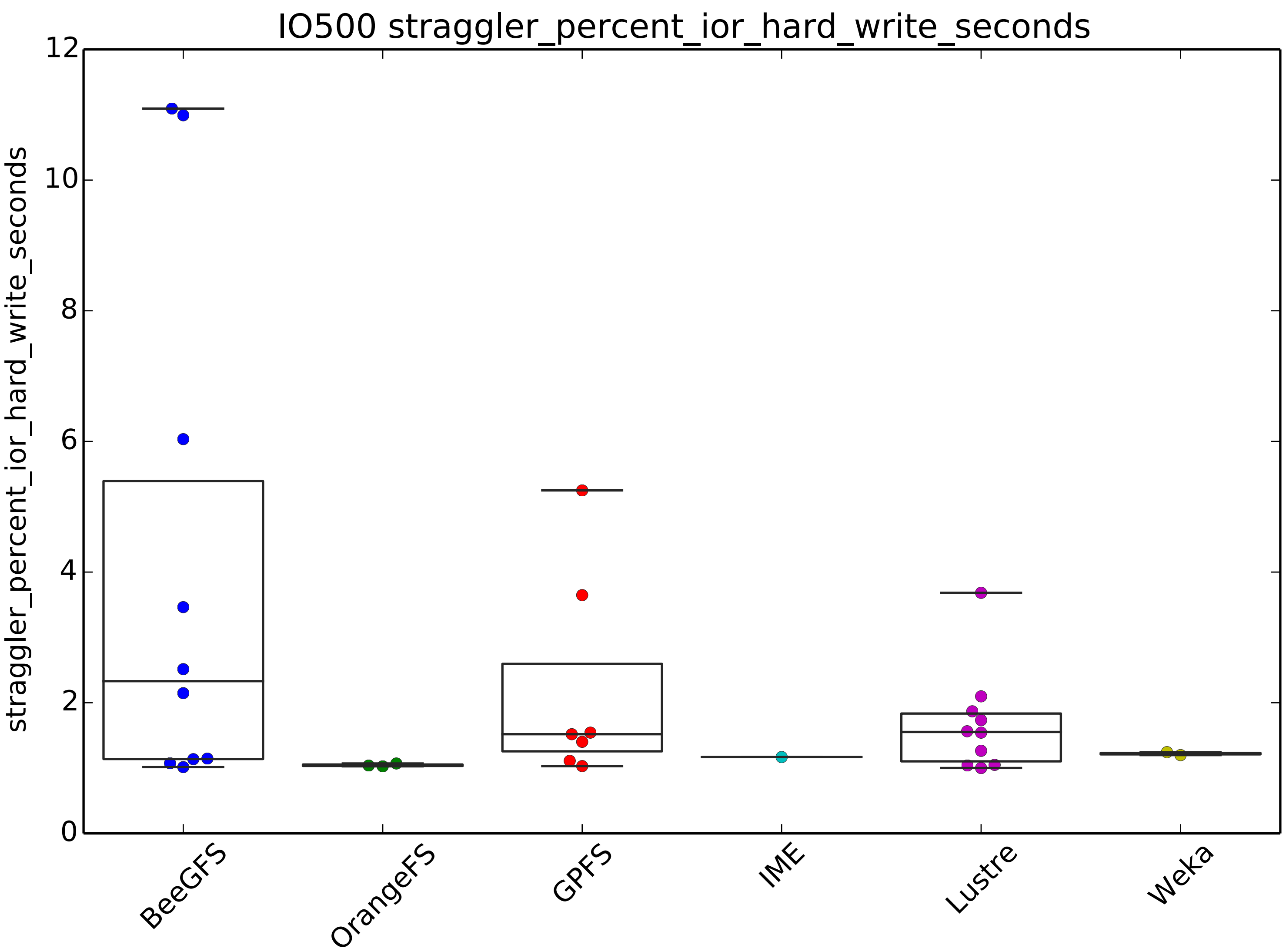
Remember a value of 1 might indicate a balanced system.

OrangeFS looks great for balance in ior_hard_write.

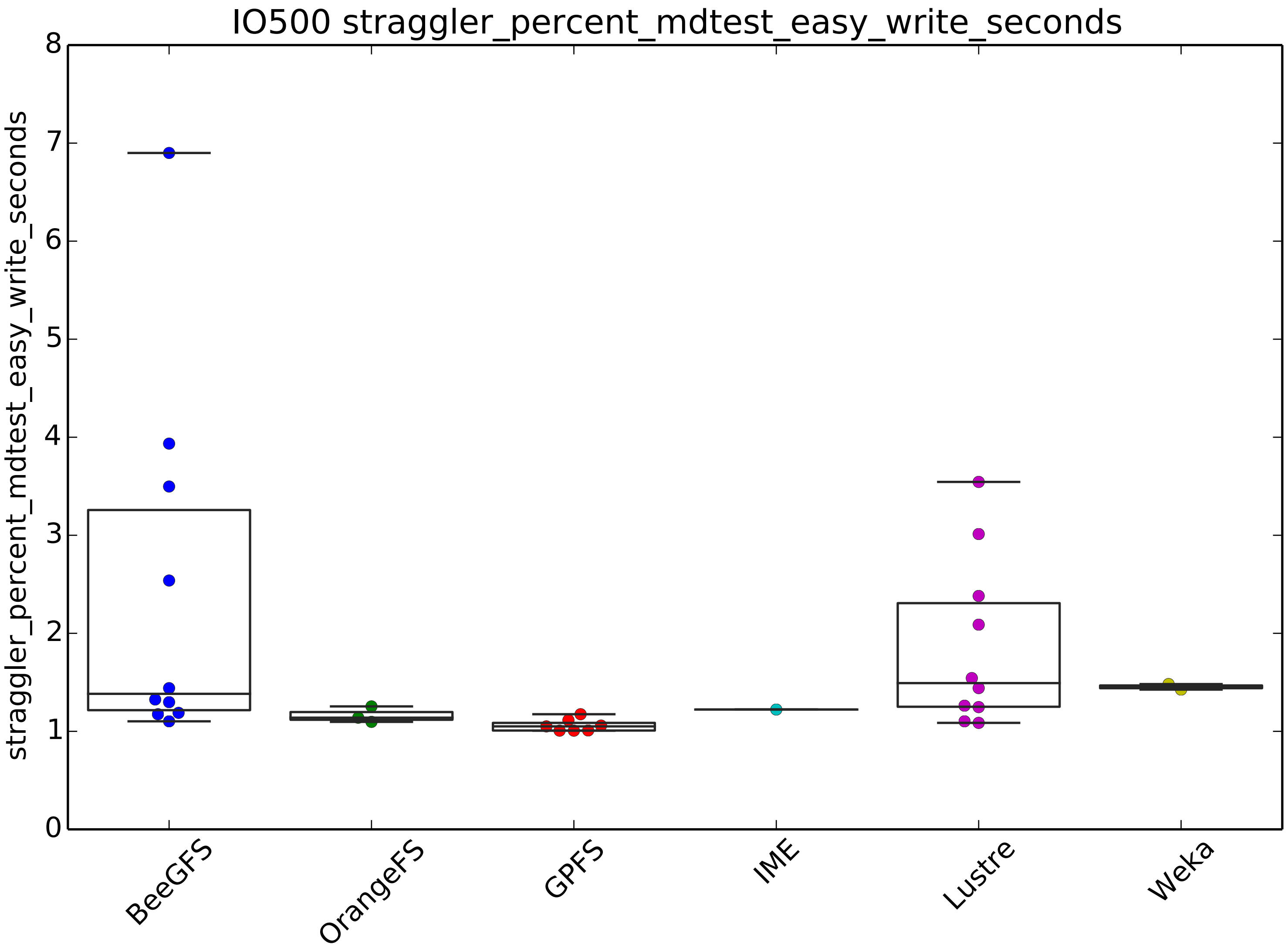
Of course the other consideration is total amount of work done.

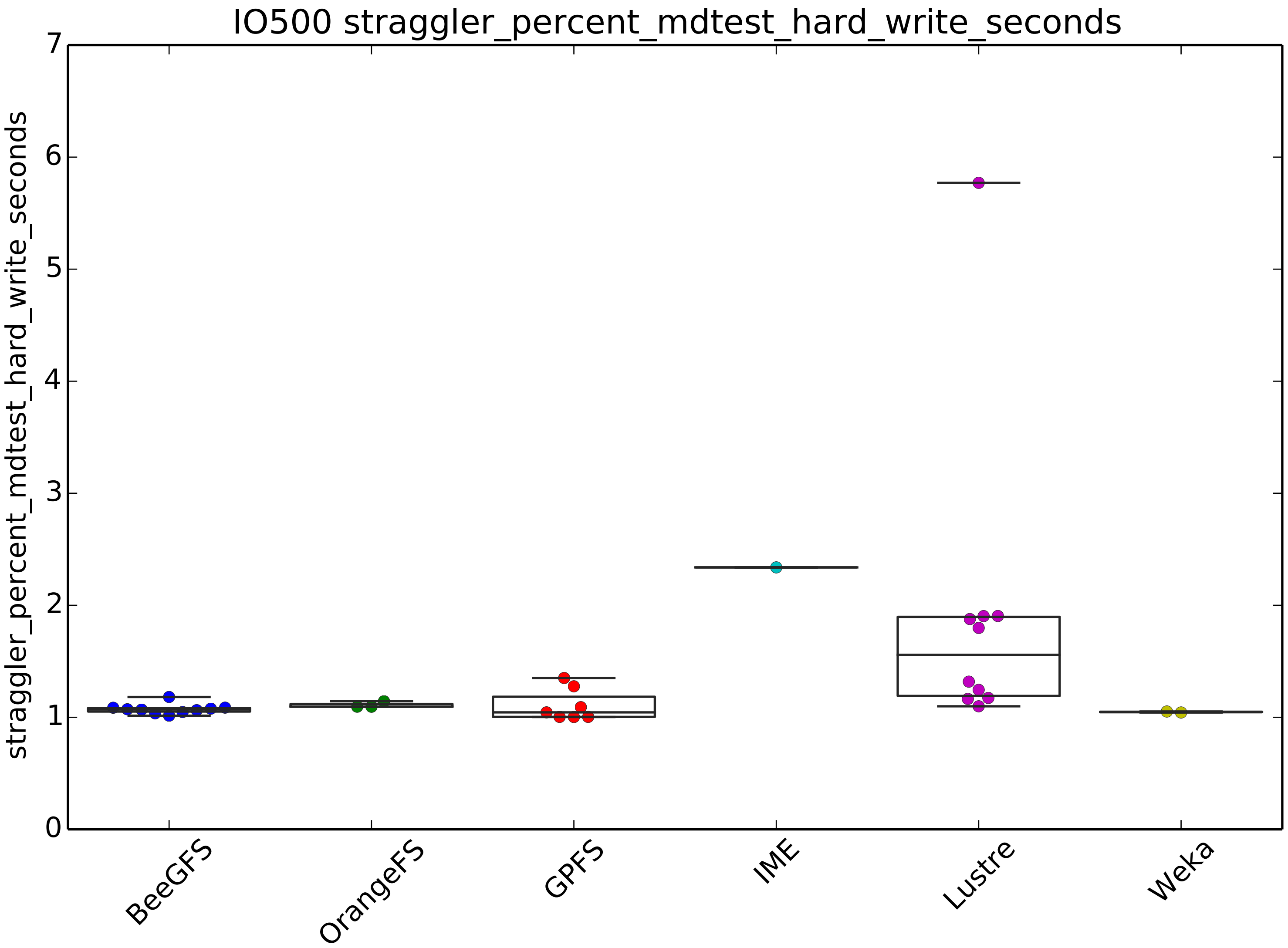
If ior_hard_write is very challenging, very little work might be done and therefore there might be less variance than with ior__easy_write.

However, there are some points here where the straggler effect is much higher than in any of the ior_easy_write results…



OrangeFS and GPFS look pretty balanced for mdtest_easy_write.





Visually comparing to the previous graph, it looks like mdtest_hard is less likely to incure imbalance overall than is mdtest_easy. Across the board, the straggler effect is low here.

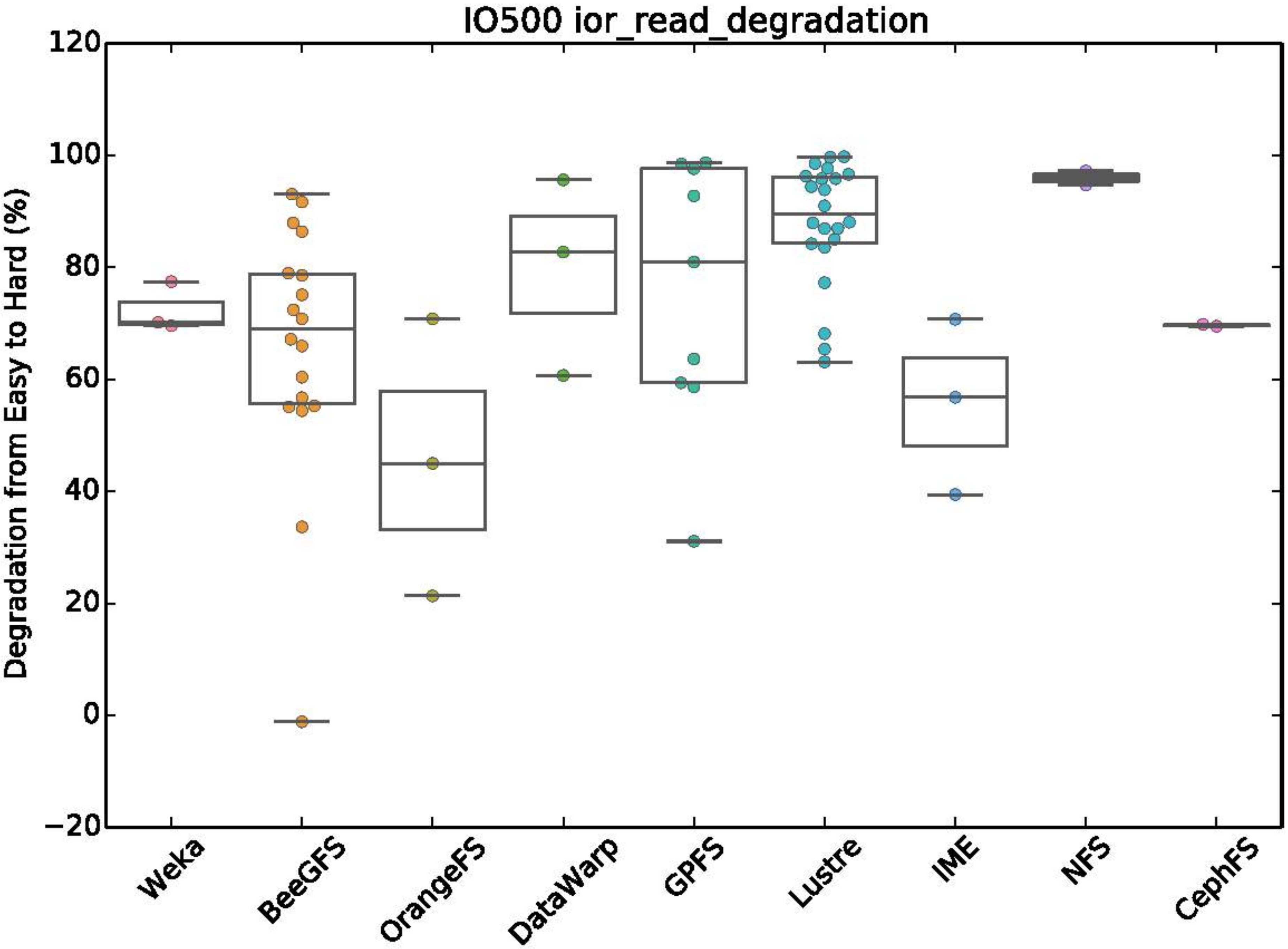
Degradation: “Measuring the Bounding Box of Expectation”

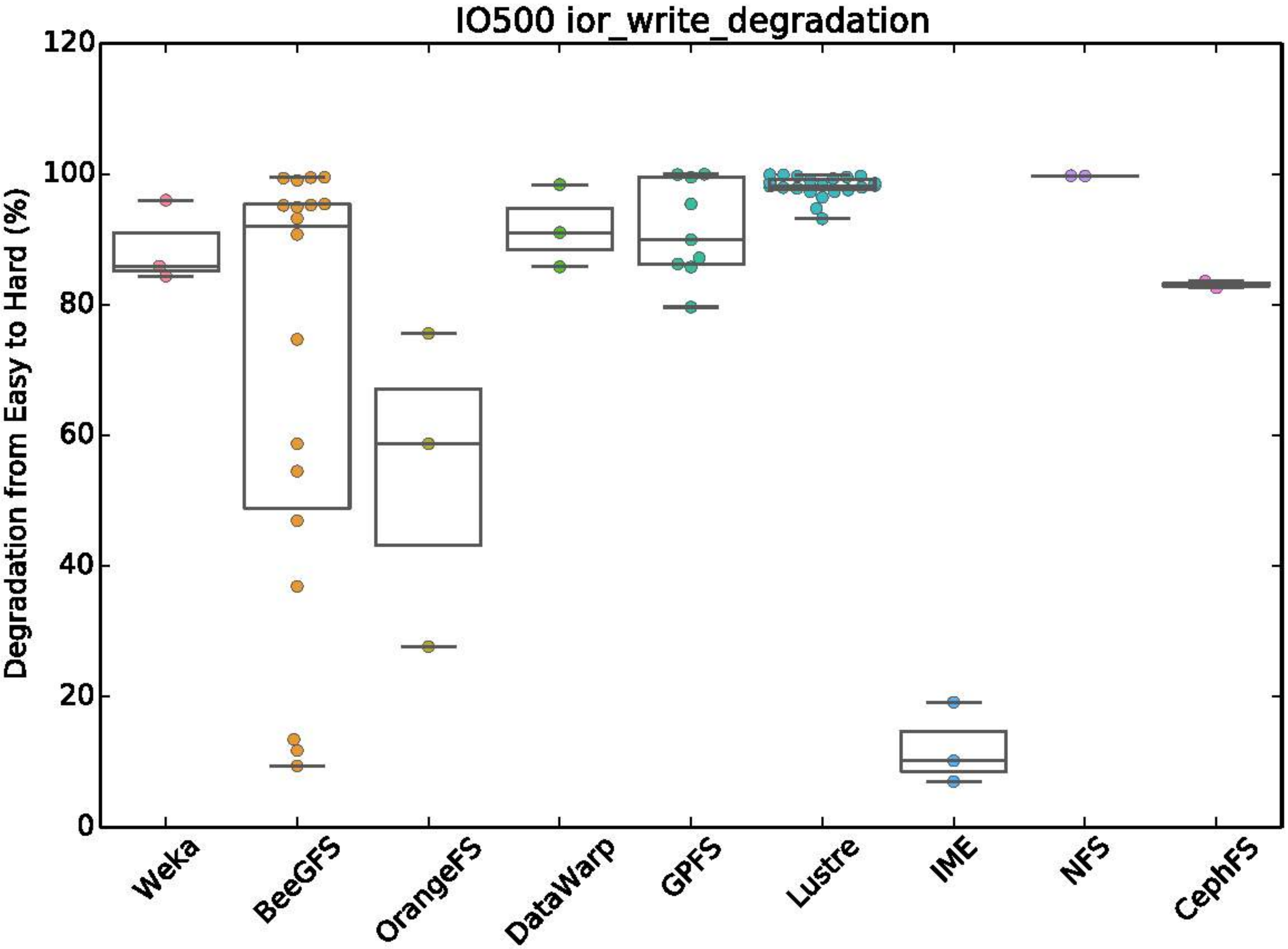
- As mentioned in the intro, one possible virtue of a system is to have a small “Bounding Box of Expectation”
- In other words, the difference between hard and easy is minimal such that every user of a system has a reasonable expectation of performance within a small bounds
- This also minimizes the need to tune applications

- In the following graphs, we therefore look at the degradation from easy to hard

In these graphs, 0% means that the hard score was identical to the easy score. A high value means that the hard score lost amount amount of the possible performance as measured by easy.

This graph shows degradation from ior_easy_read to ior_hard_read. Because IOR opens with O_RDONLY, there shouldn't be much locking and degradation here should be overall low. That is not what we see however.





As expected, the log-structure approach in IME means very little performance is lost from ior_easy_write to ior_hard_write.

BeeGFS has an interesting spread. Some great results at the bottom! But some bad ones at the top. The next slide tries to figure out why…

Why Does BeeGFS have such a large degradation spread for ior_write?

```
mysql> select (ior__hard_write-ior__easy_write)*-1/ior__easy_write*100 as Degradatio
n,information__system as System,information__client_nodes as CN,concat("Data on ",io
500_info_data_storage_type," metadata on ",io500_info_metadata_storage_type) as Stor
age, io500_info_filesystem_version as Version from io500 where information__filesyst
em rlike 'beegfs' order by ior__hard_write/ior__easy_write;
+--------------+-------------------+------+------------------------------+---------+
| Degradation  | System            | CN   | Storage                      | Version |
+--------------+-------------------+------+------------------------------+---------+
| 99.520636984 | Clemson BeeGFS    |   16 | Data on HDD metadata on SSD  | 7       |
| 99.462218576 | Clemson BeeGFS    |   10 | Data on HDD metadata on SSD  | 7.1     |
| 99.382203781 | Clemson BeeGFS    |   16 | Data on HDD metadata on SSD  | 7       |
| 99.047069015 | Clemson BeeGFS    |   16 | Data on HDD metadata on SSD  | 7       |
| 95.409140279 | Data Accelerator  |  184 | Data on xxx metadata on xxx  | xxx     |
| 95.263437999 | Seislab           |   24 | NULL                         | NULL    |
| 95.200525970 | JURON             |    8 | NULL                         | NULL    |
| 94.952681388 | Palmetto          |   32 | Data on HDD metadata on SSD  | 7.1     |
| 93.206521739 | Palmetto          |   32 | Data on HDD metadata on SSD  | 7.1     |
| 90.759075908 | Palmetto          |   32 | Data on HDD metadata on SSD  | 7.1     |
| 74.691358025 | Clemson BeeGFS    |   16 | Data on HDD metadata on SSD  | 7       |
| 58.713136729 | Palmetto          |   32 | Data on HDD metadata on SSD  | 7.1     |
| 54.497354497 | Palmetto          |   16 | Data on HDD metadata on SSD  | 7.1     |
| 46.930422920 | Clemson BeeGFS    |   32 | Data on HDD metadata on SSD  | 7       |
| 36.877828054 | Palmetto          |   16 | Data on HDD metadata on SSD  | 7.1     |
| 13.424657534 | Palmetto          |   10 | Data on HDD metadata on SSD  | 7.1     |
| 11.738148984 | Palmetto          |   48 | Data on HDD metadata on SSD  | 7.1     |
|  9.385704499 | Palmetto          |   32 | Data on HDD metadata on SSD  | 7.1     |
+--------------+-------------------+------+------------------------------+---------+
18 rows in set (0.07 sec)
```

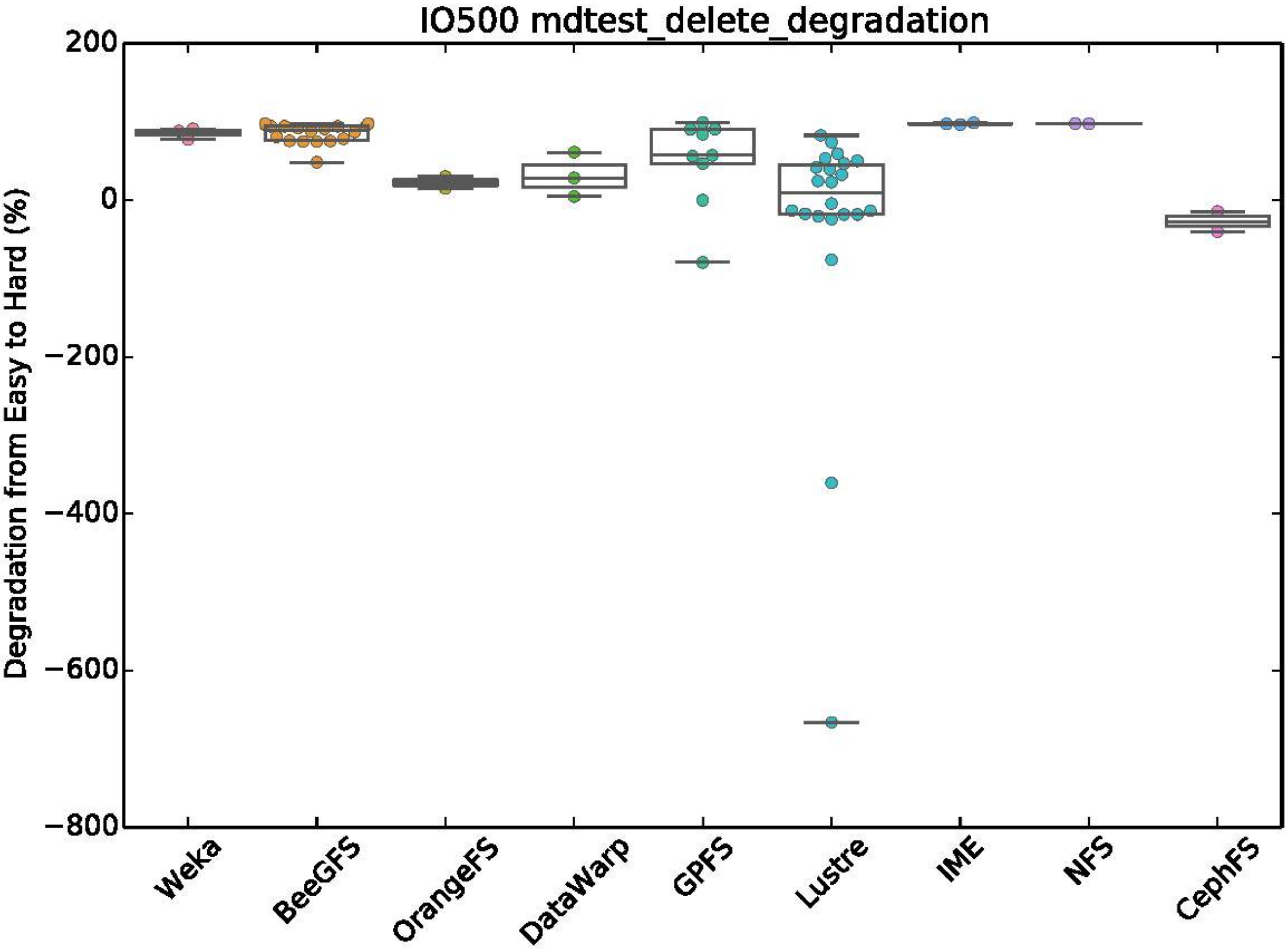
Large Spread!
Why?!?!

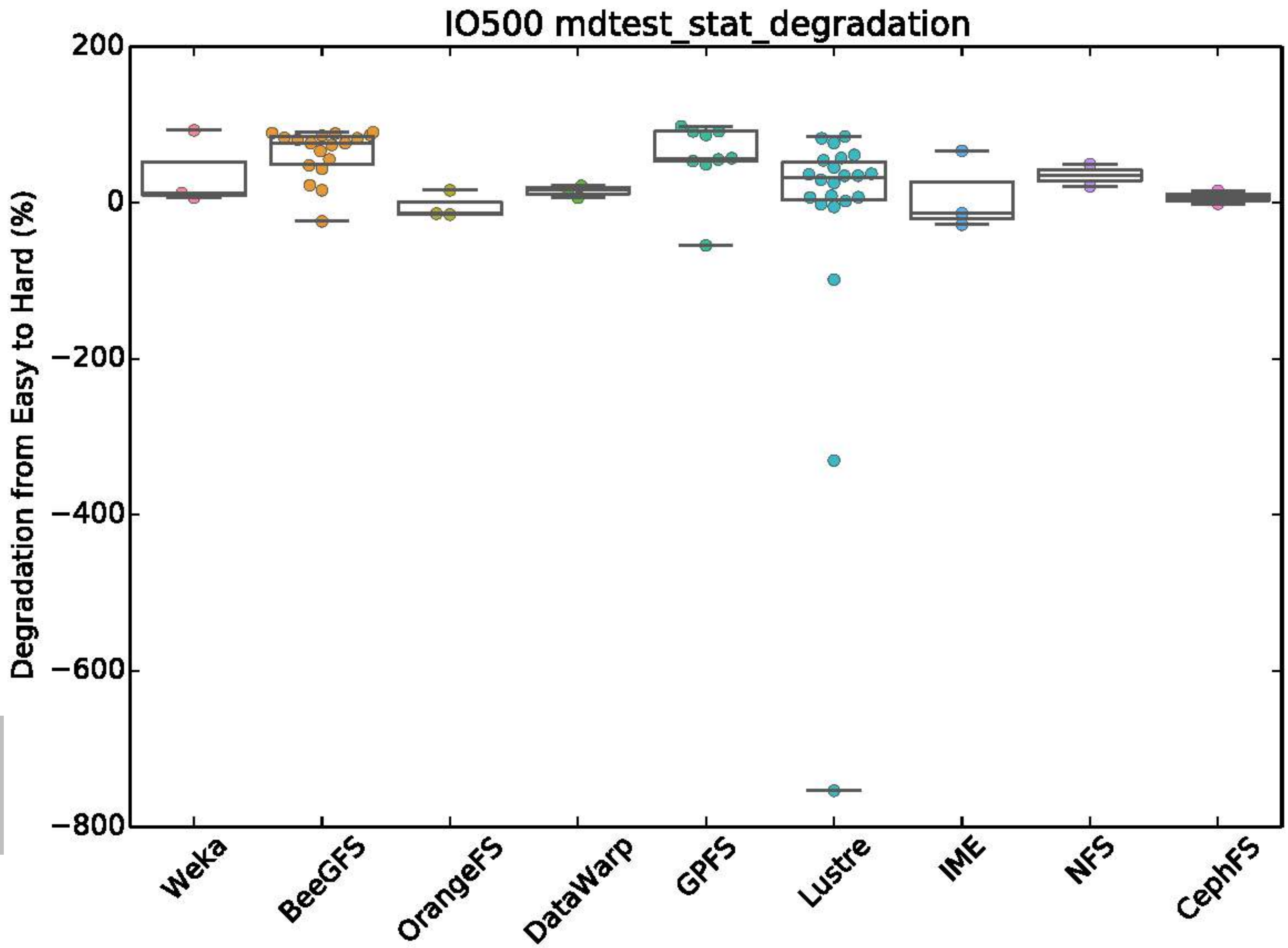
I don't know.

Reminder: 0% is the “target” here as it indicates no loss from easy to hard. 100% would be the worst case and indicates that hard is infinitely worse than easy.

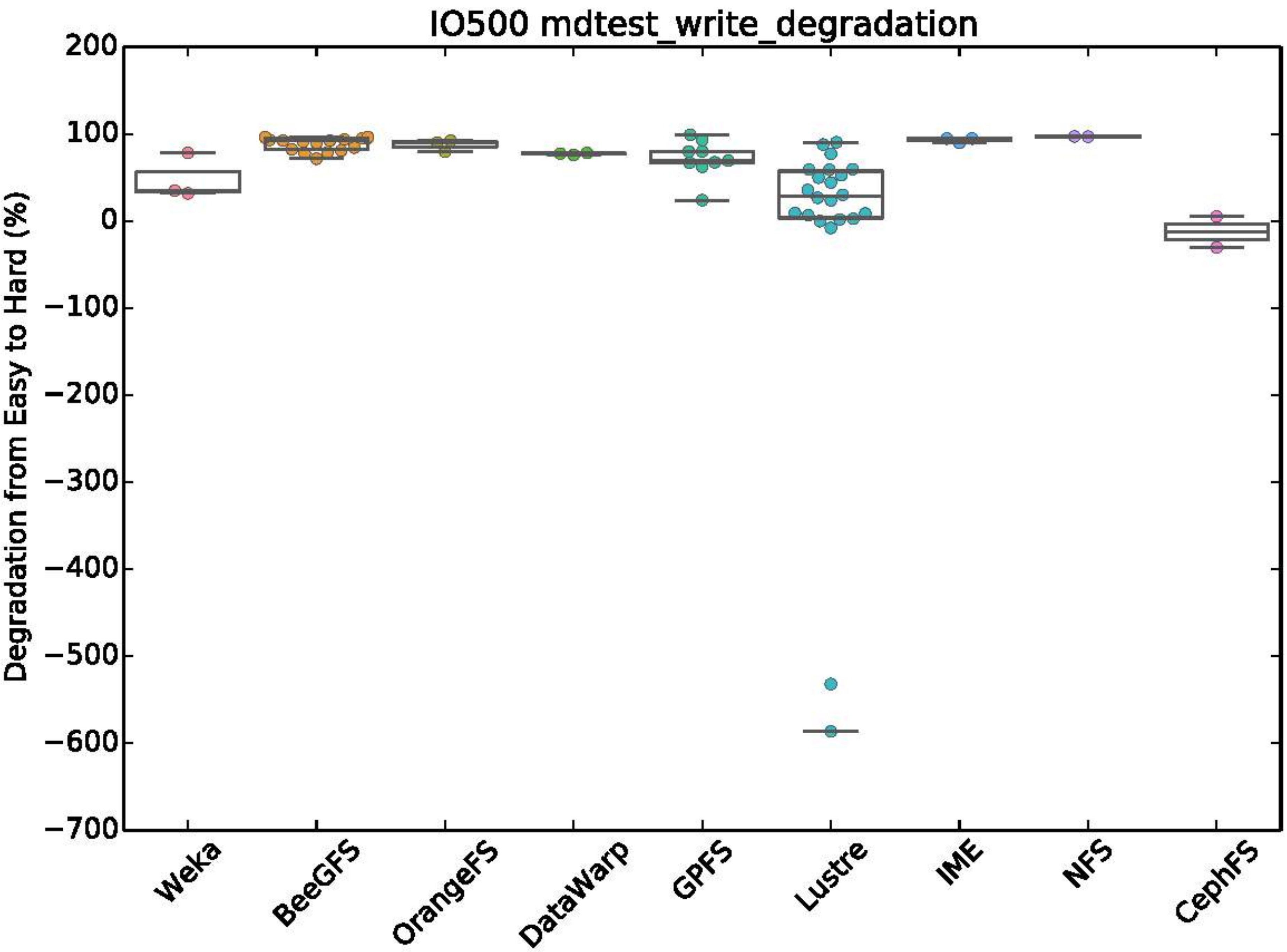
A negative result is unexpected and means that hard did better than easy.

The Lustre results from Cambridge do have metadata results where hard is better because they tuned hard to use DNE2 but did not tune easy so easy results only used one MDS/MDT.





Same explanation about Cambridge Lustre as on previous slide.

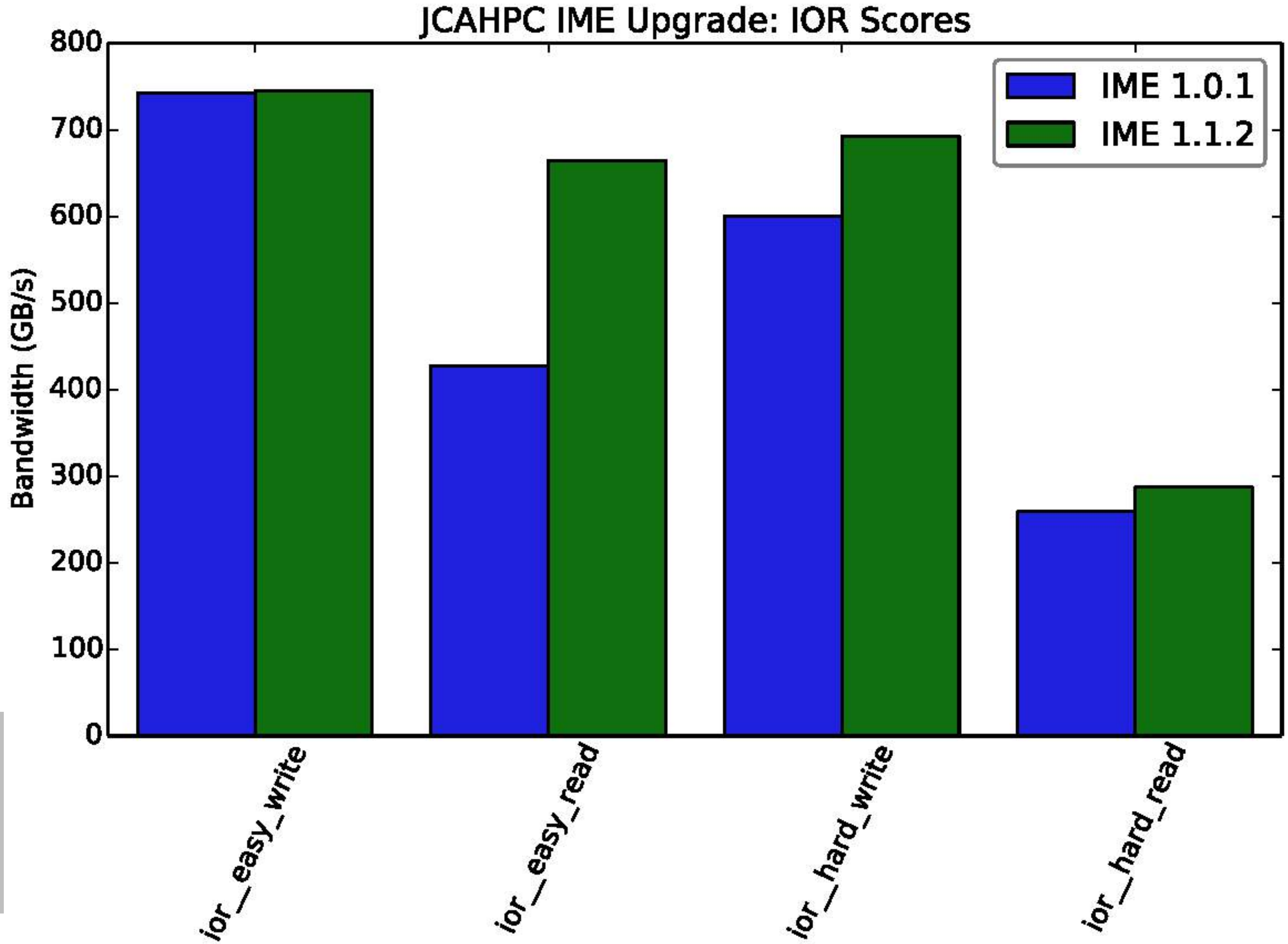


IO500

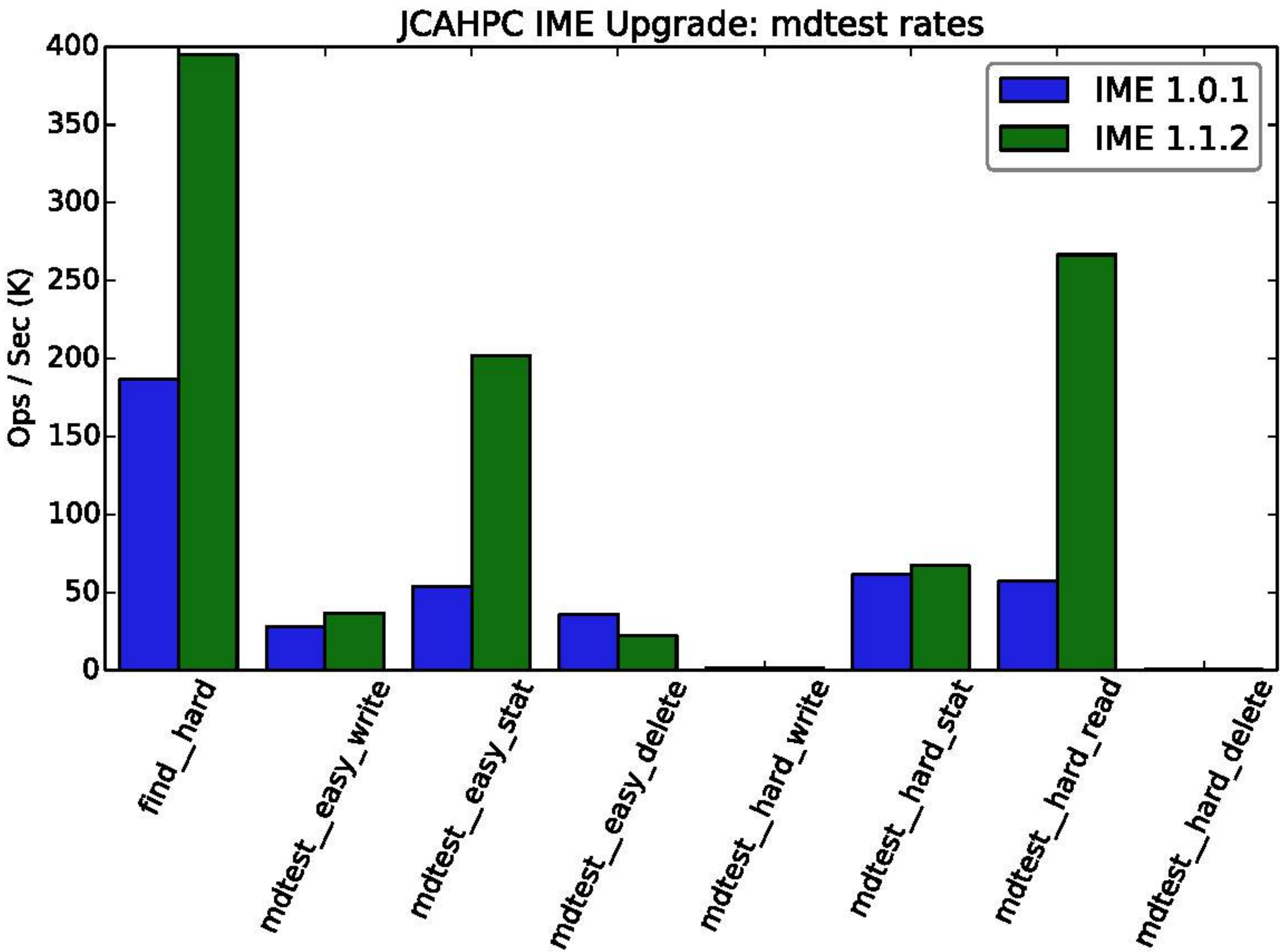
Same explanation about Cambridge Lustre as on previous slide.

JCAHPC

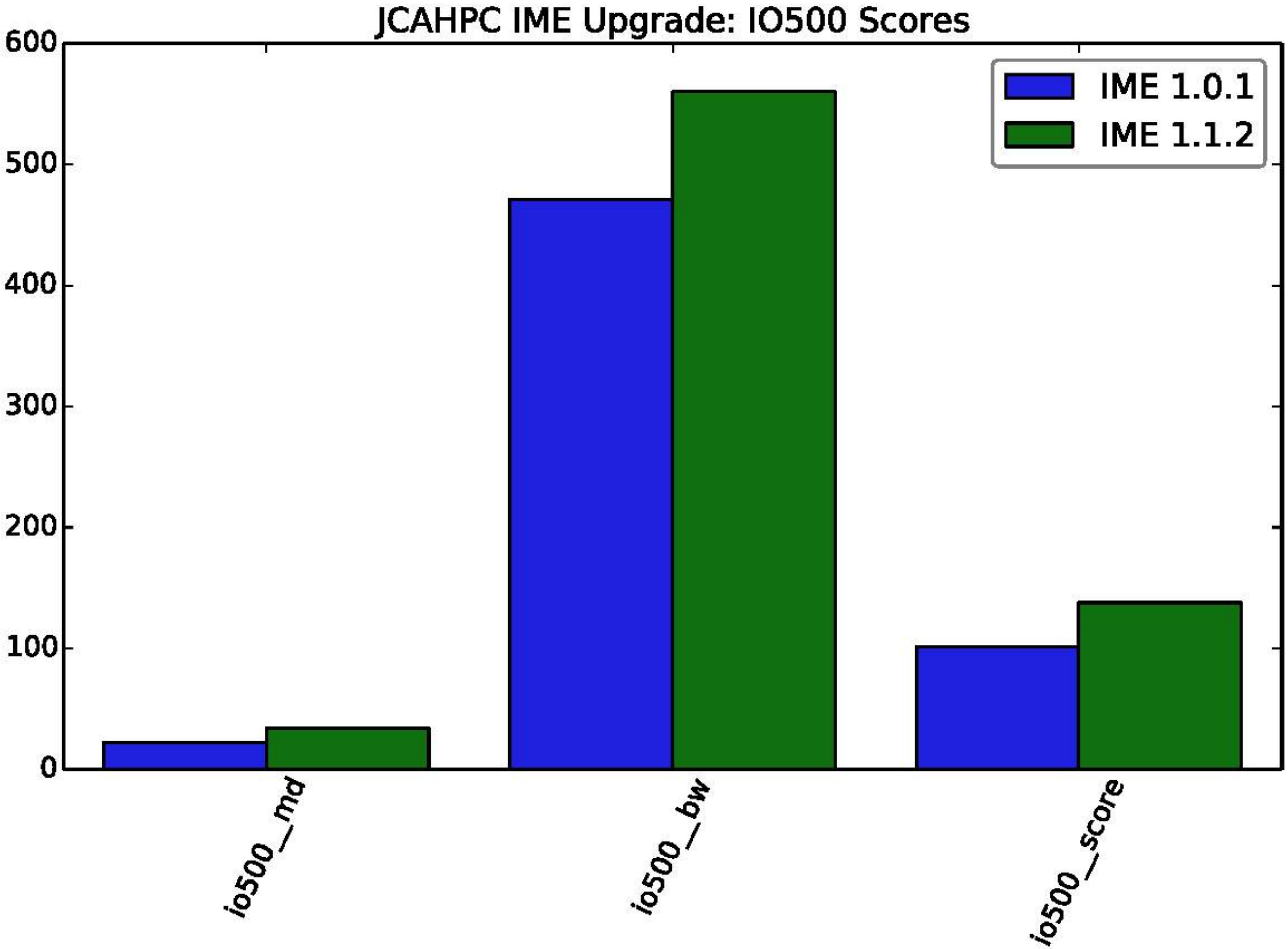
- Use IO500 for regression testing or to check whether a software upgrade actually improves the system



Upgrading IME improves bandwidth! This is apples-apples hardware comparison.



Upgrading IME improves metadata except delete got worse. Something to debug . . . Better in 1.2?

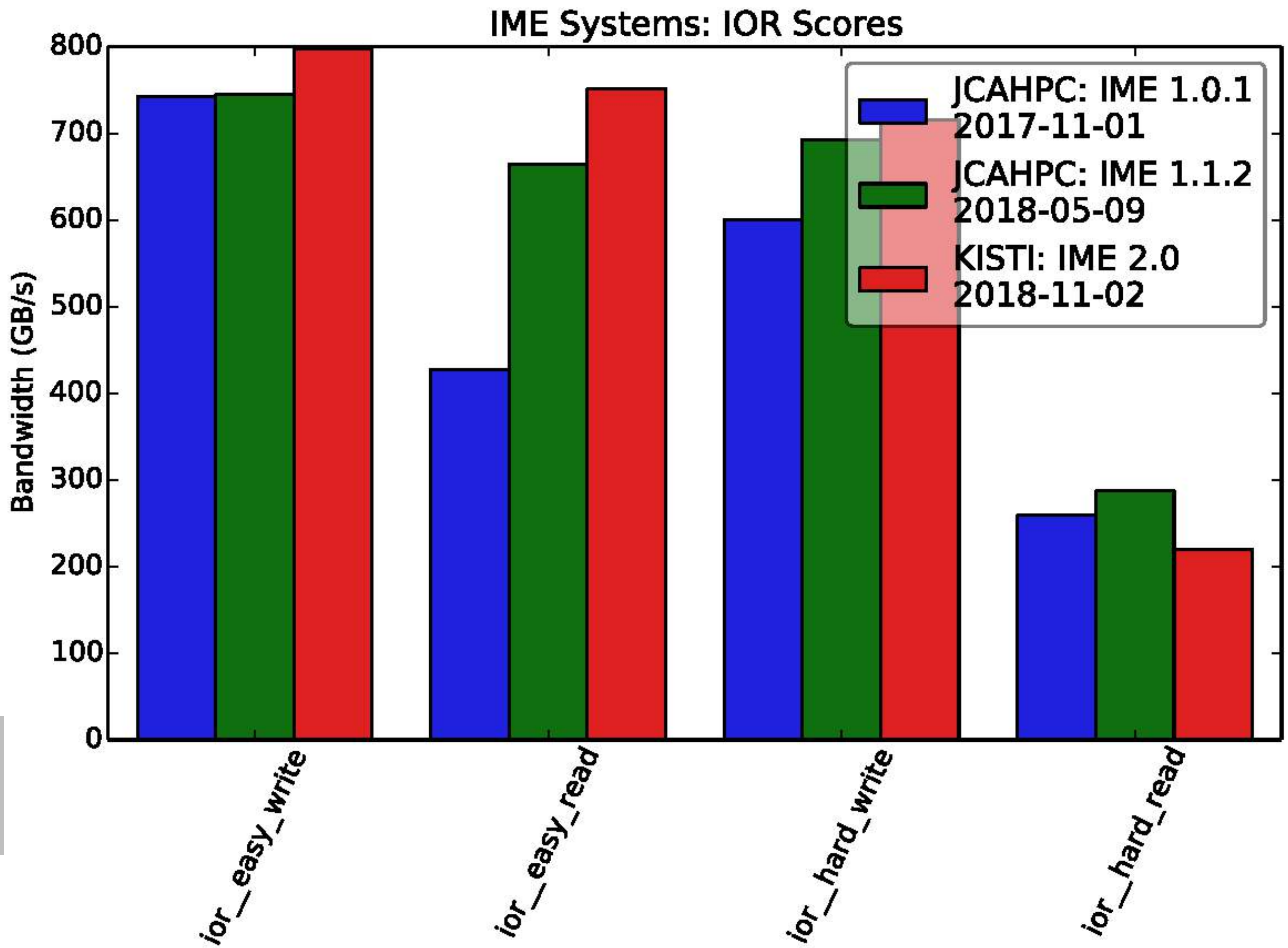


As expected, overall scores went up.

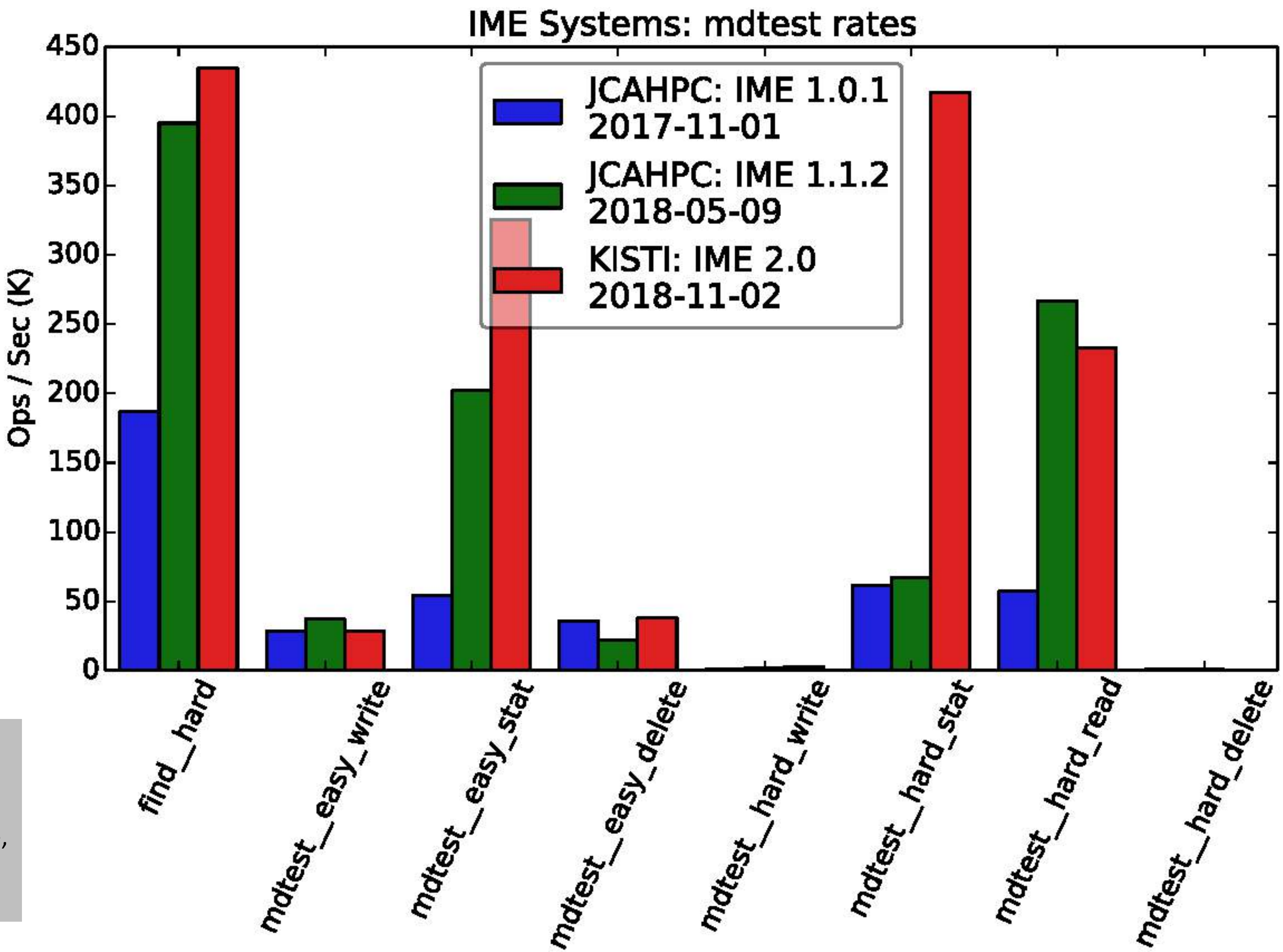
Good job IME developers!

IME Results

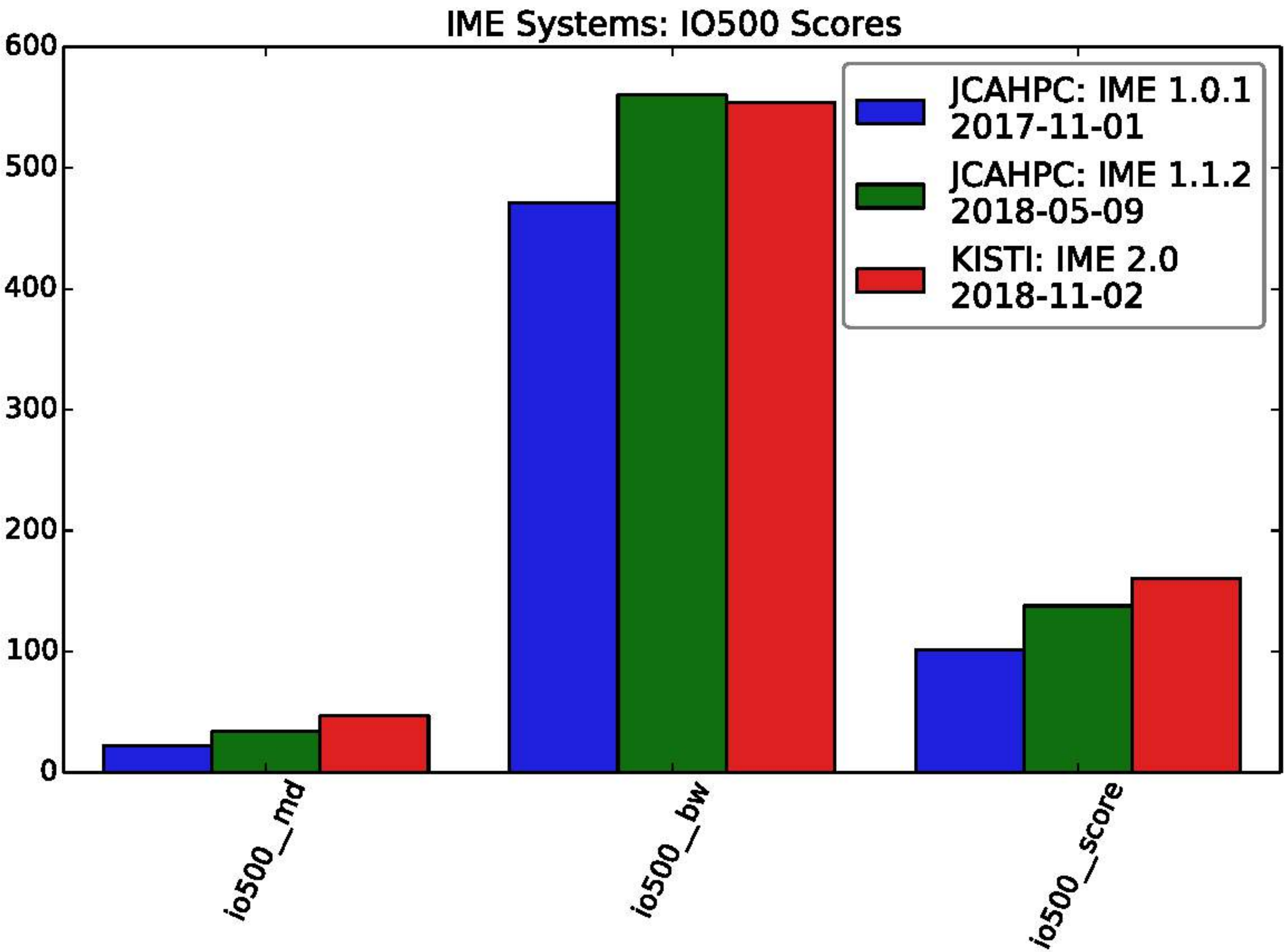
- To see whether IME 1.2 further improves over IME 1.1.2, we can compare the KISTI result to the two JCAHPC results
- Unfortunately this is different hardware so comparison might be tricky



Except for ior_hard_read, it looks like the expected trend.



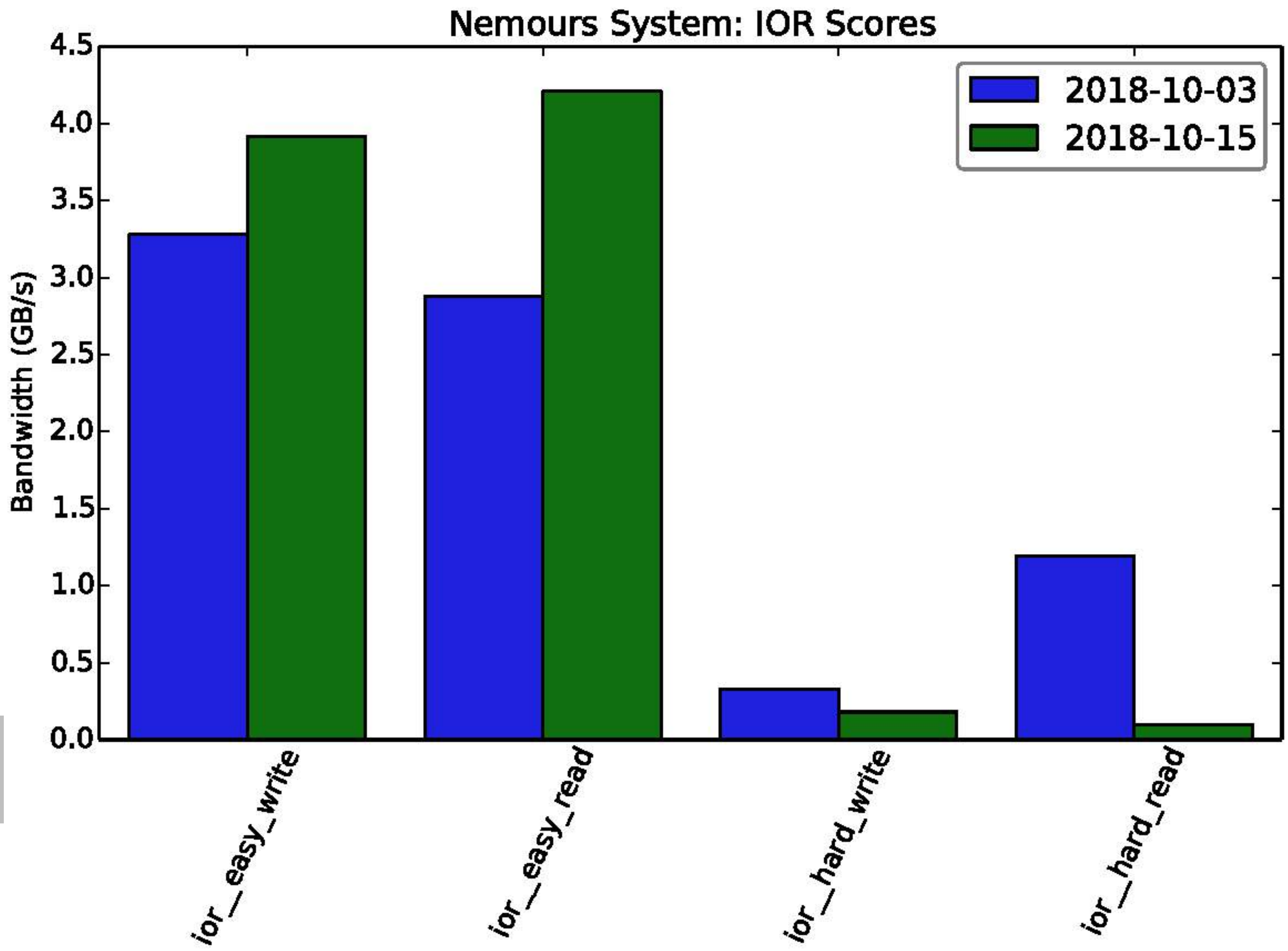
Except for mdtest_hard_read and mdtest_easy_write, it looks like the expected trend.



Bandwidth went slightly down but the improvement in metadata was enough to improve the overall score.

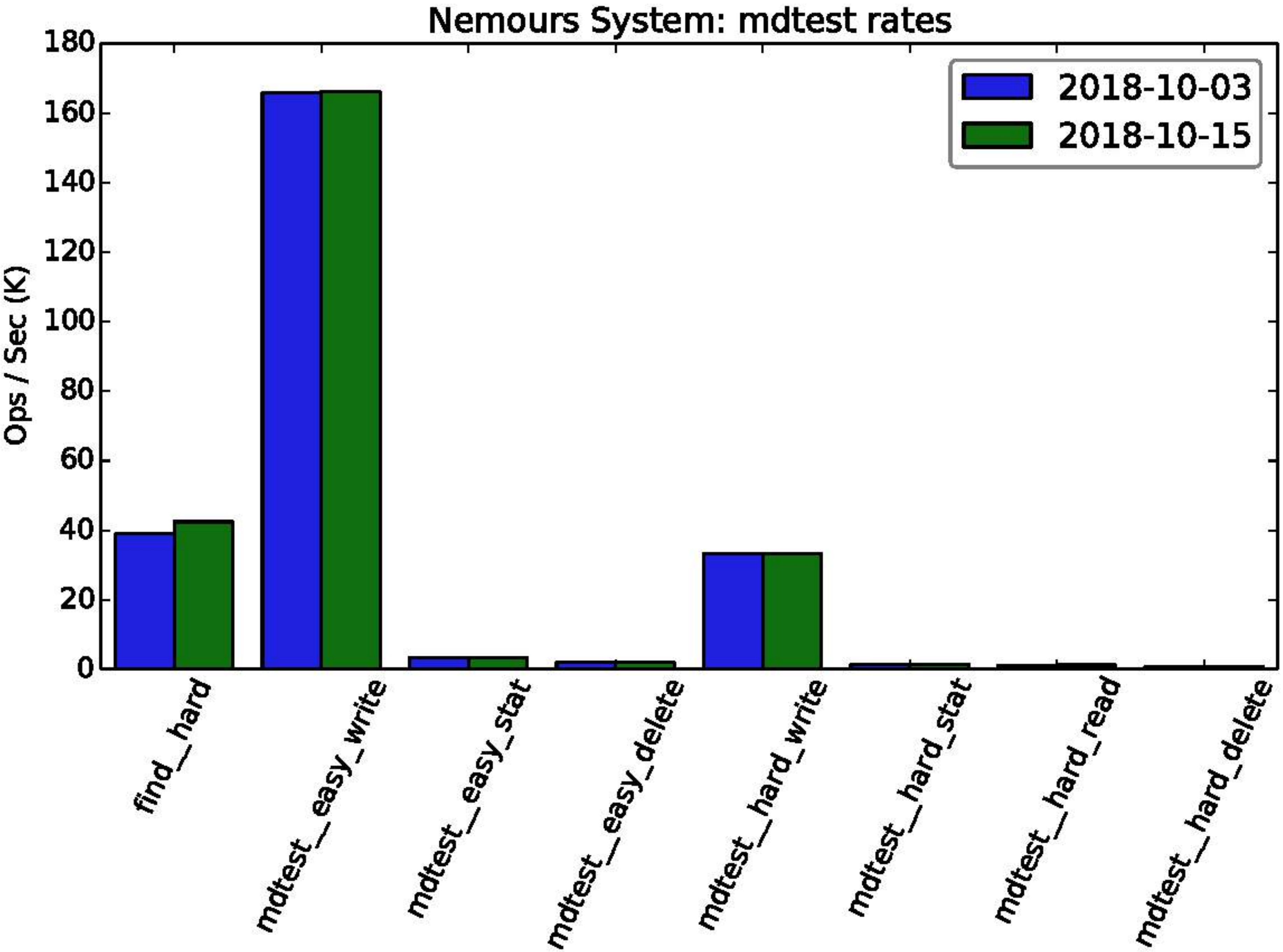
Regression Testing at Nemours

- Nemours also used IO500 to test a system before and after an upgrade.
- Unfortunately the overall performance went down. Why?
- Noise during testing?
- The upgrade was actual a downgrade?
- The test only used one client so perhaps that's more of a measure of the particular client node than of the overall file system?
- More data needed!

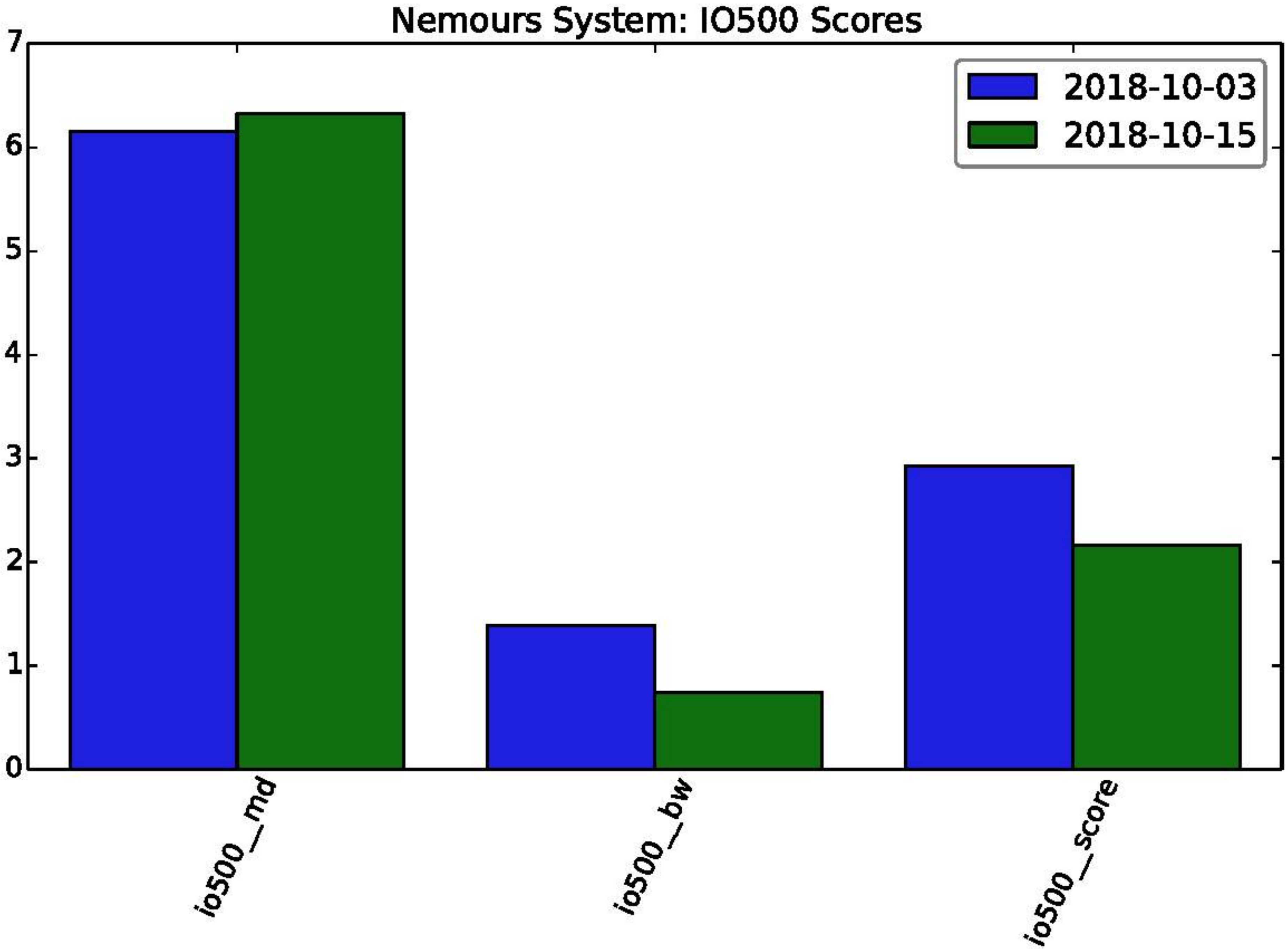


IO500

IOR easy got better but hard got worse.



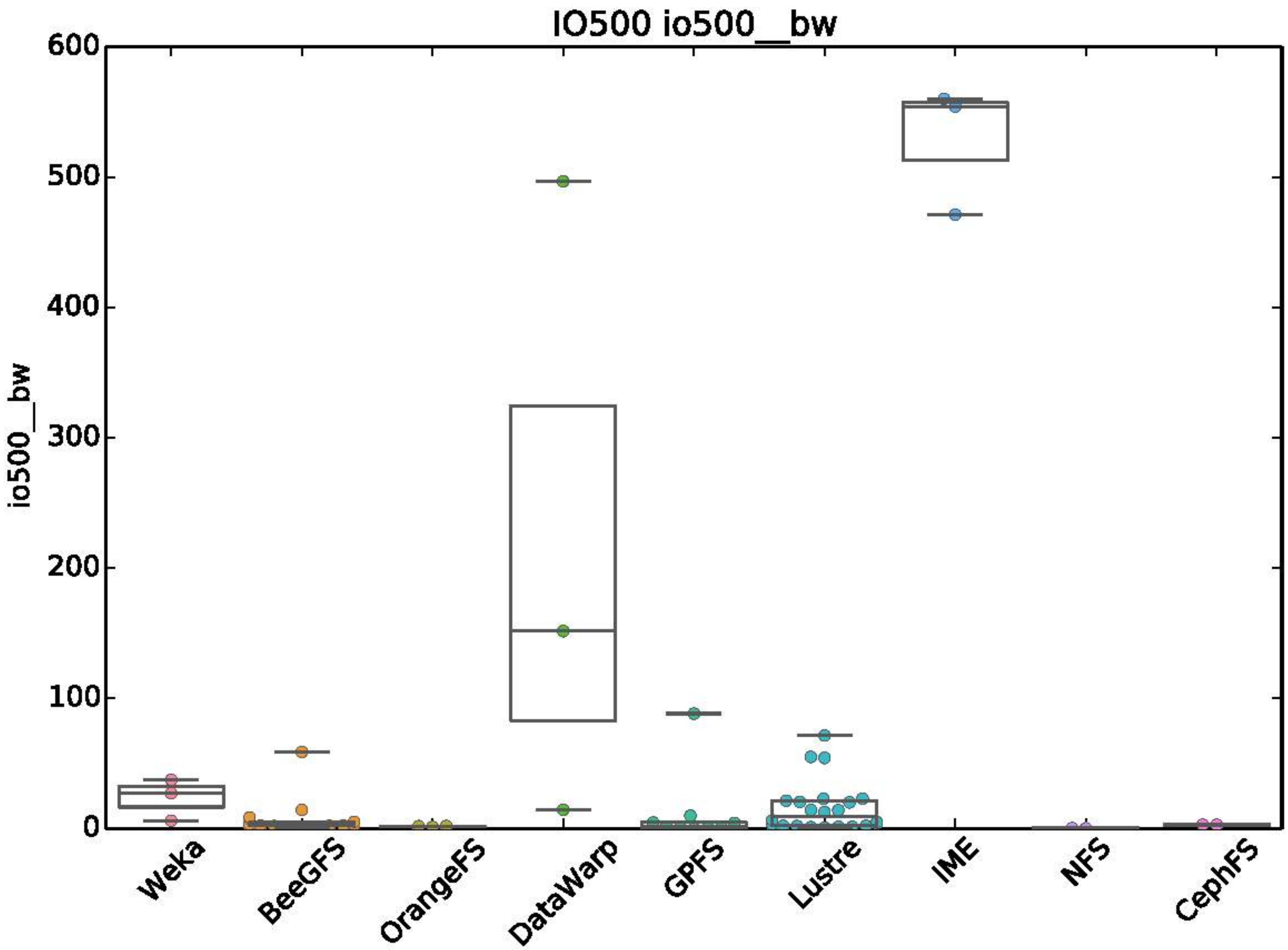
Not much change in metadata.

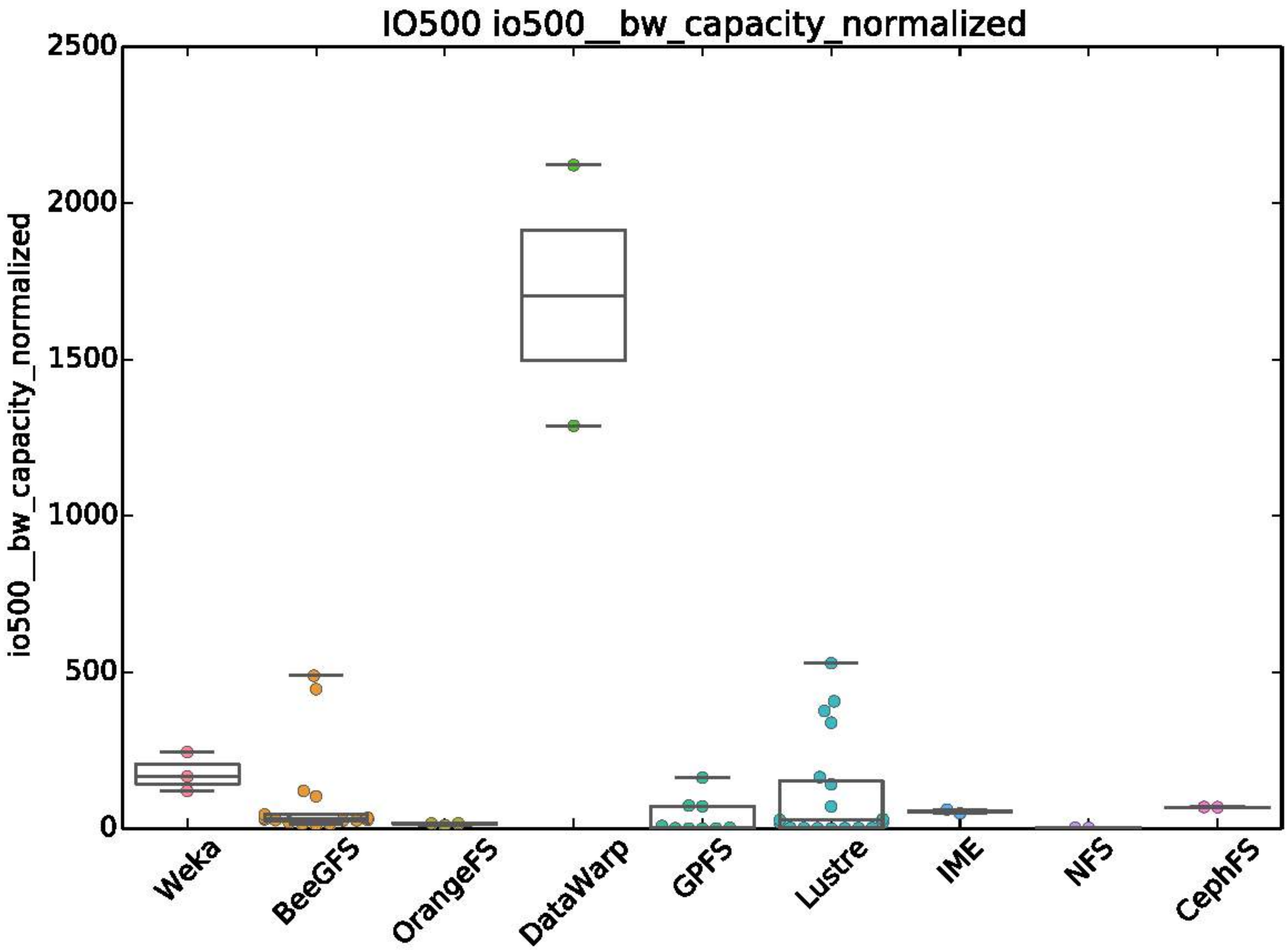


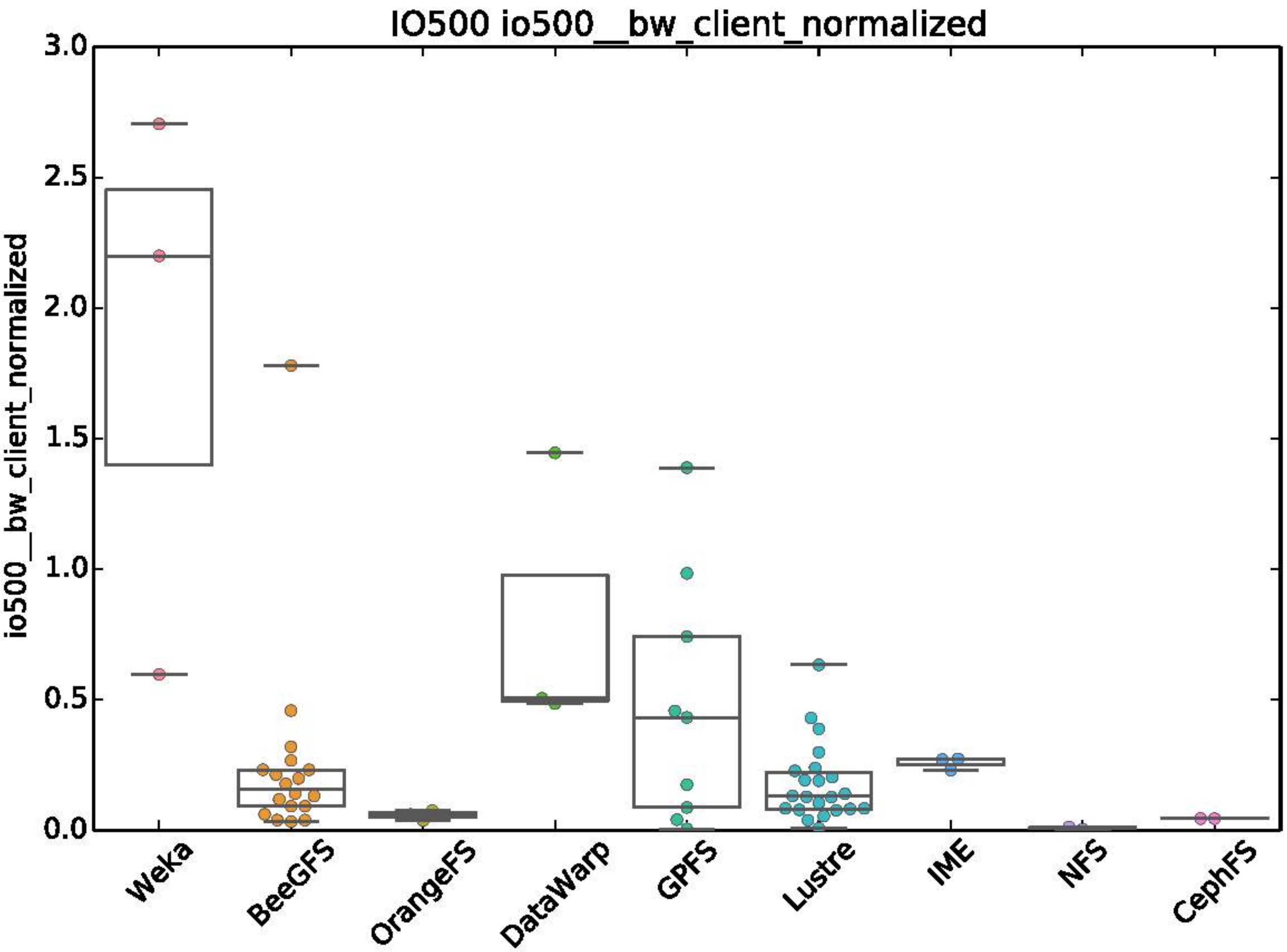
A mixed bag but overall the score went slightly down.

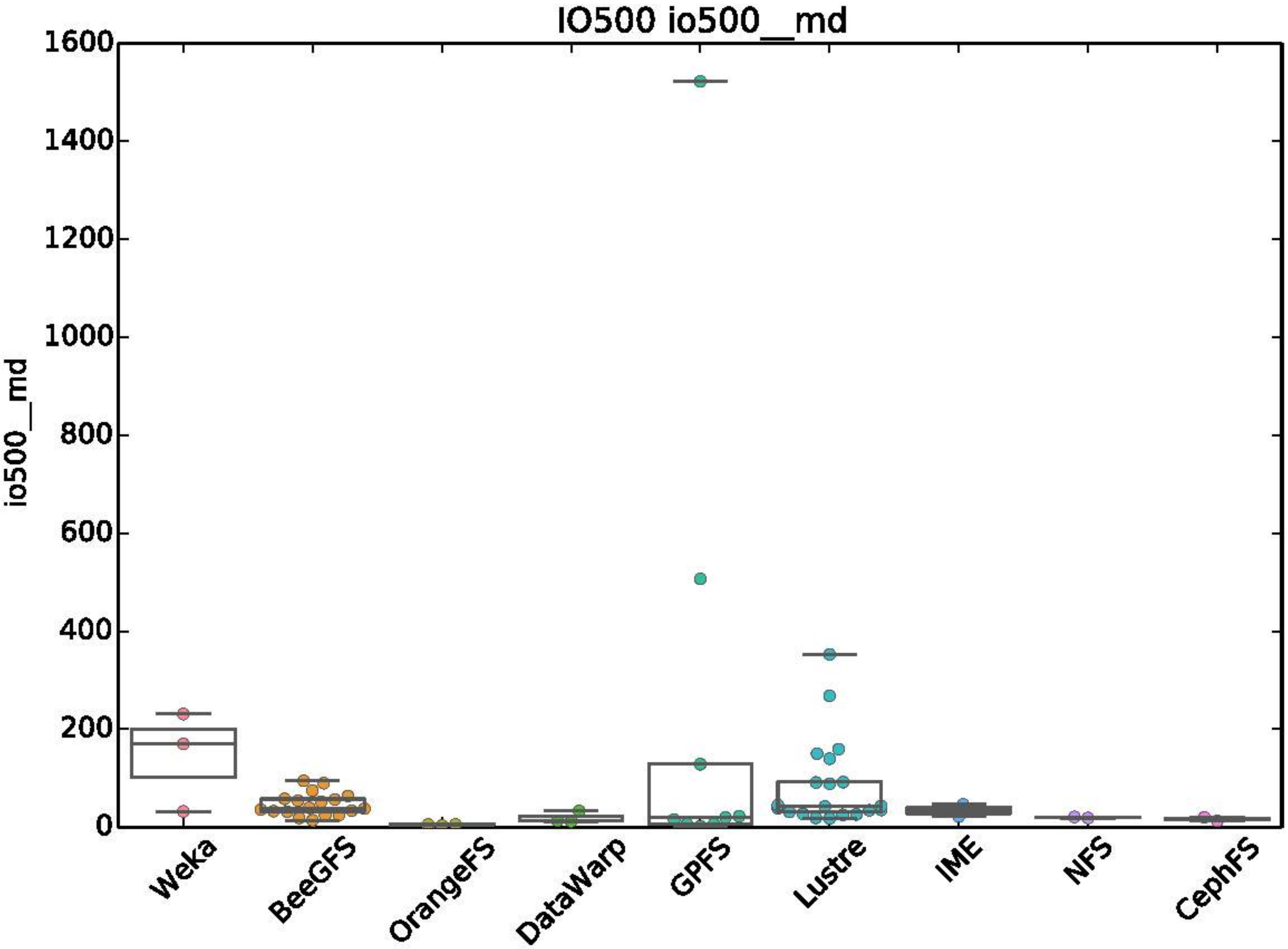
Tons and Tons of Boxplots

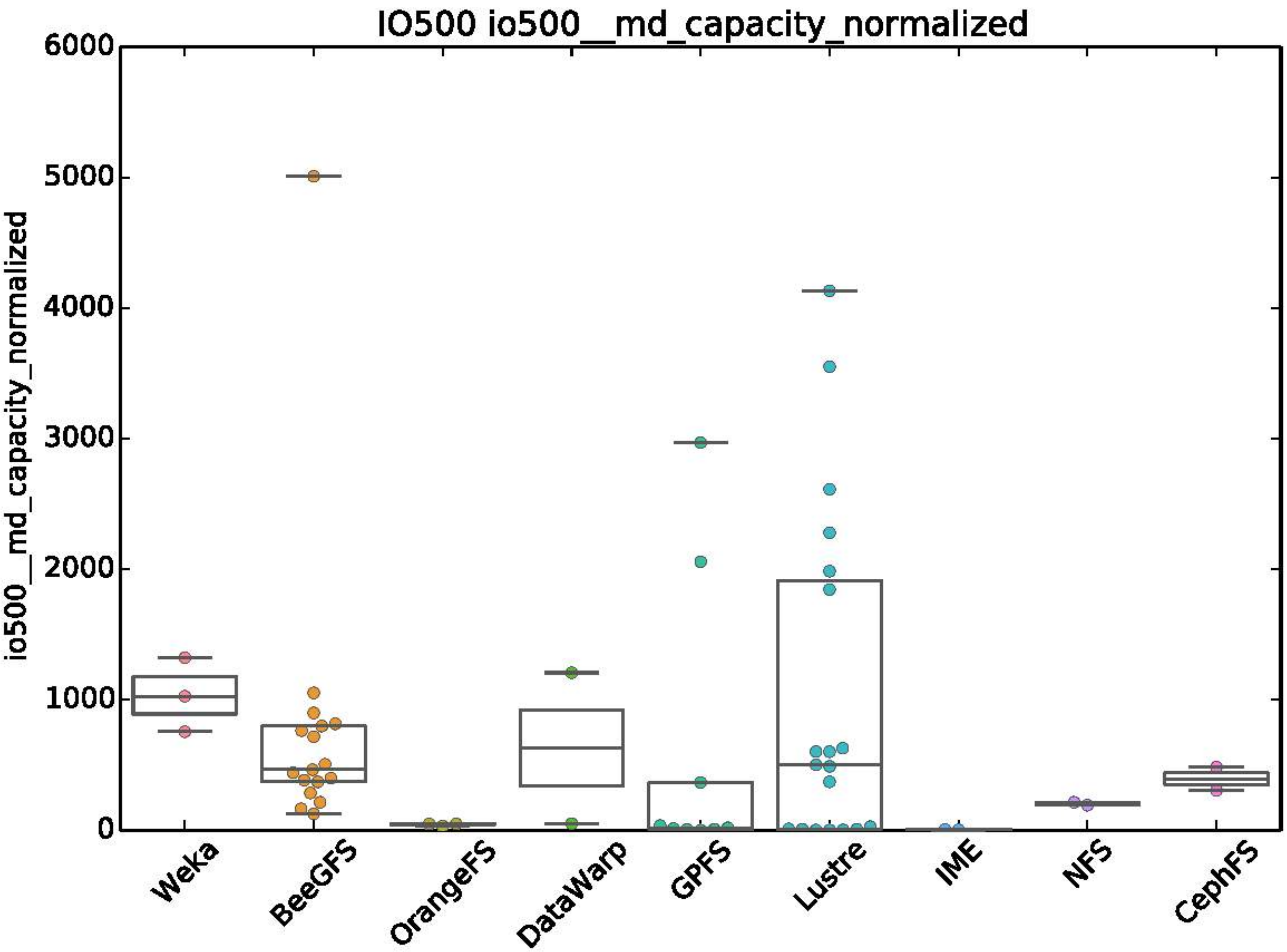
- For each metric, all the results grouped by file system
- Also includes an attempted normalization by client count
- Also includes an attempted normalization by total capacity
  - Note total capacity is inaccurate!
  - io500.sh collects df by calling ‘df’. However, ‘df’ reports in block and different systems use a different value for block size.
  - We need to update io500.sh to pass a flag to ‘df’ to force consistency in the block size
  - More generally we need to scrape more environmental info.
  - Feel free to submit patches! ☺
- Way too many graphs to attempt analysis. Have at ye!


IO500 io500__bw
io500__bw
600
500
400
300
200
100
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

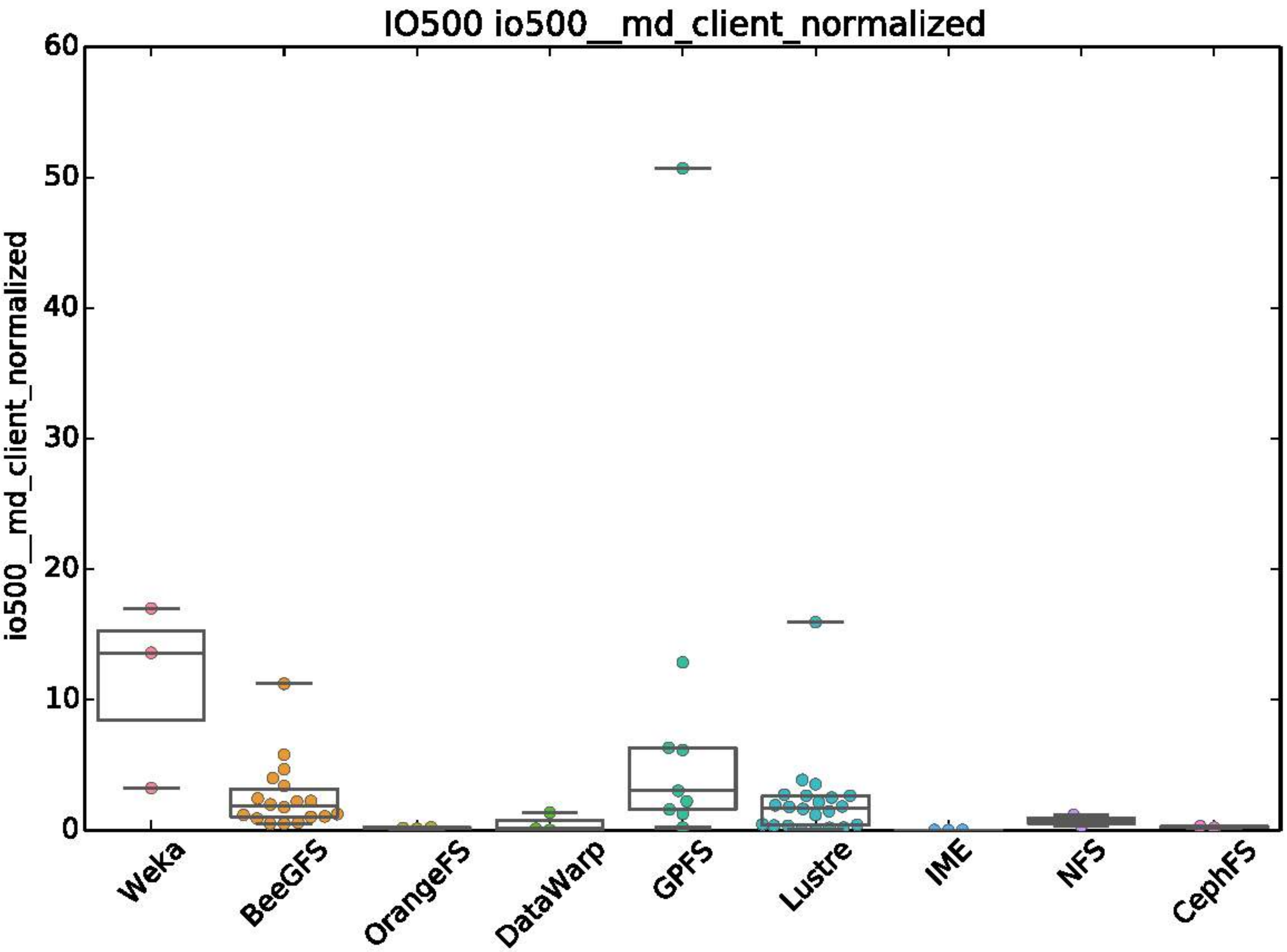

IO500 io500__bw_capacity_normalized
io500__bw_capacity_normalized
2500
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

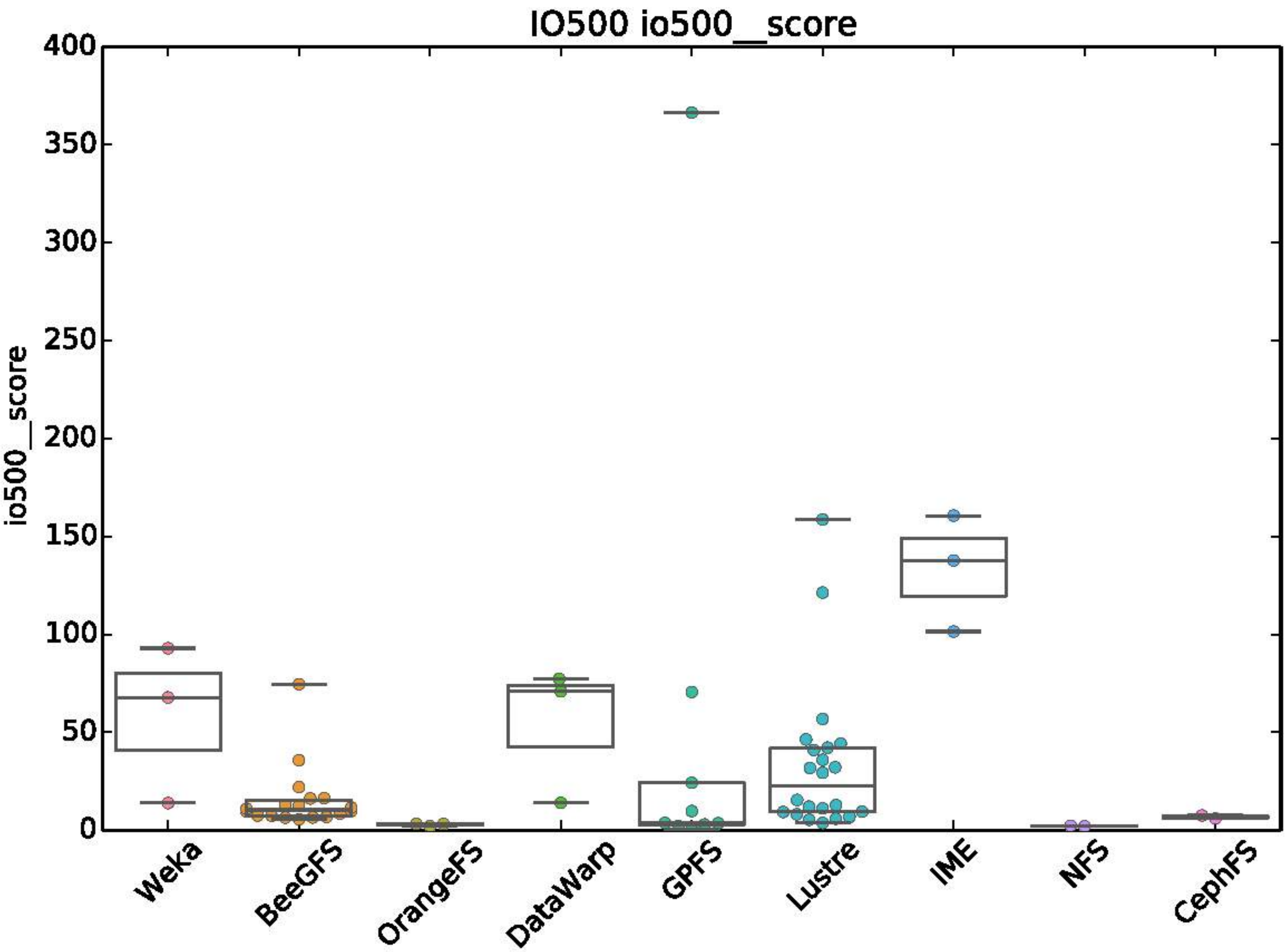

IO500 io500__bw_client_normalized
io500__bw_client_normalized
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS


IO500 io500__md
io500__md
1600
1400
1200
1000
800
600
400
200
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

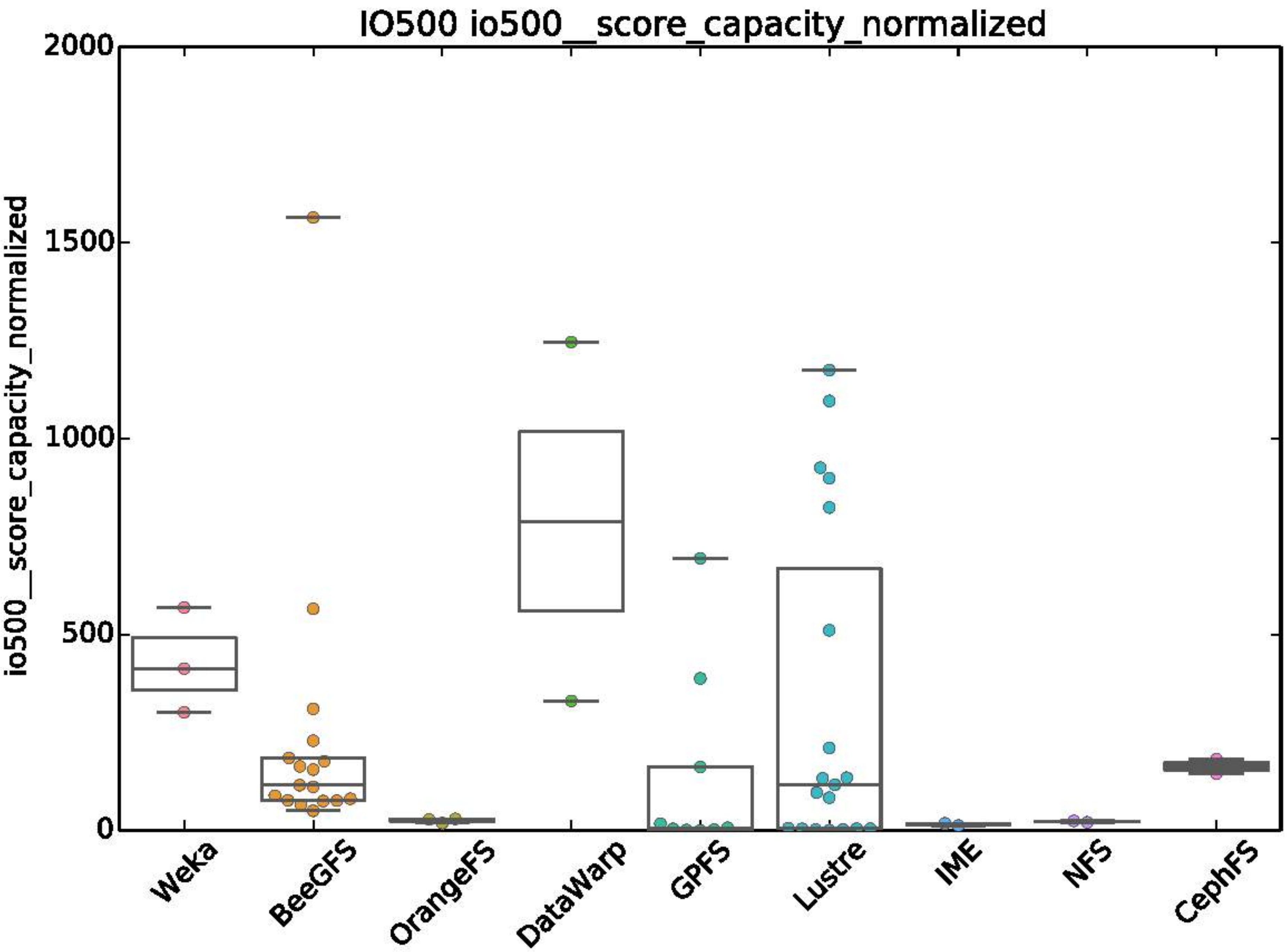


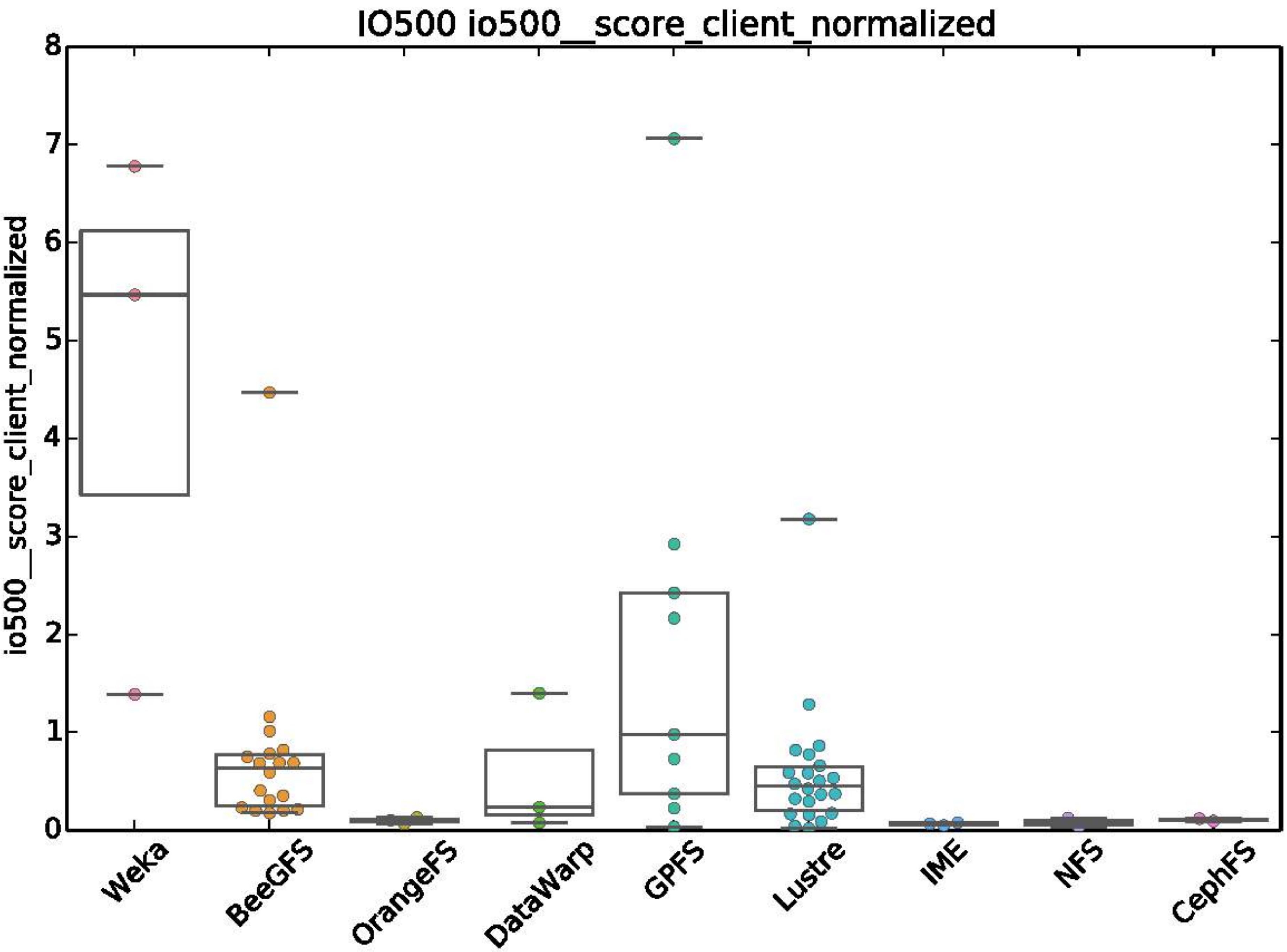
IO500 io500__md_capacity_normalized
io500__md_capacity_normalized
6000
5000
4000
3000
2000
1000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS


IO500 io500__md_client_normalized
io500__md_client_normalized
0
10
20
30
40
50
60
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

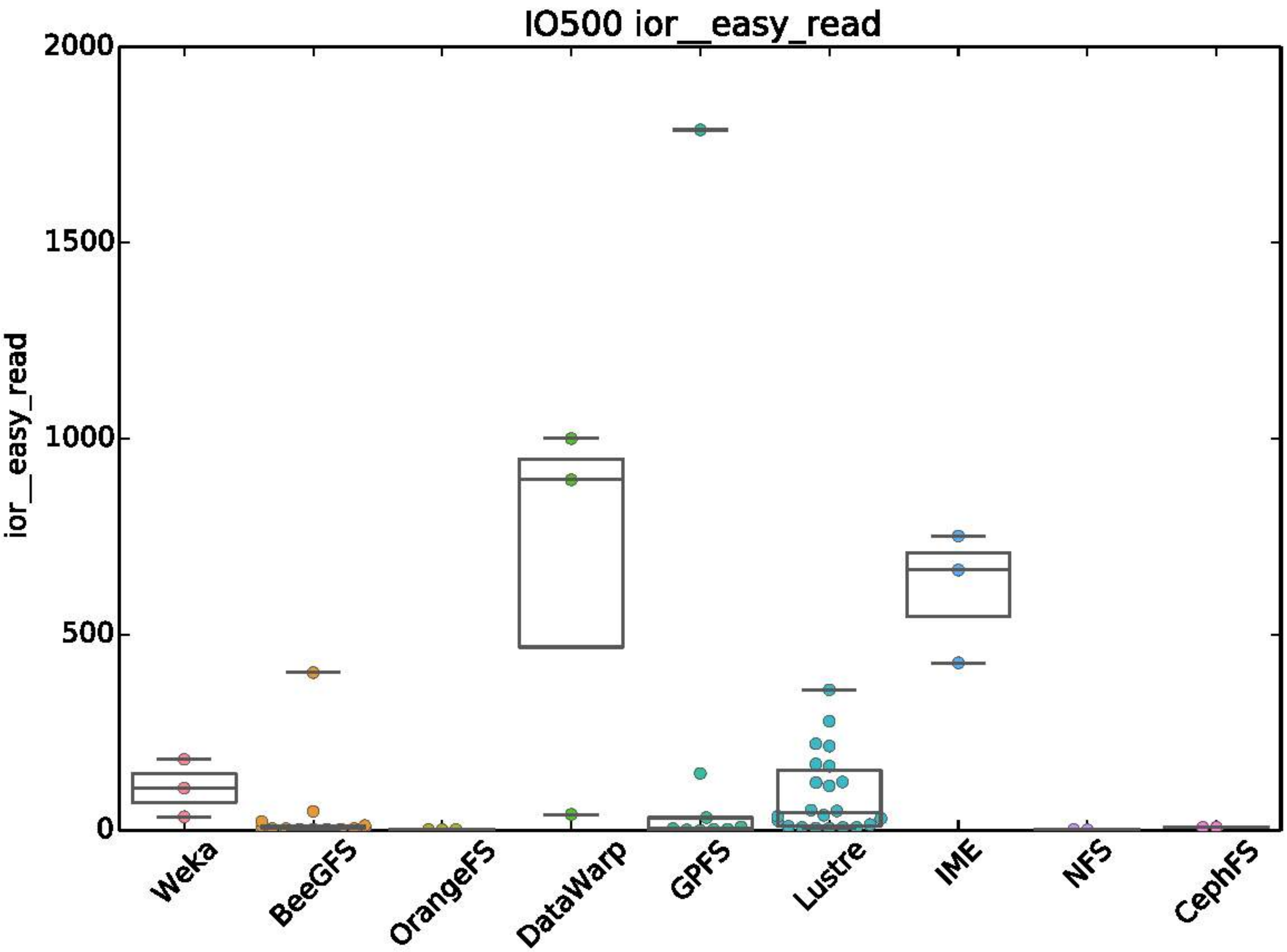

IO500 io500__score
io500__score
400
350
300
250
200
150
100
50
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

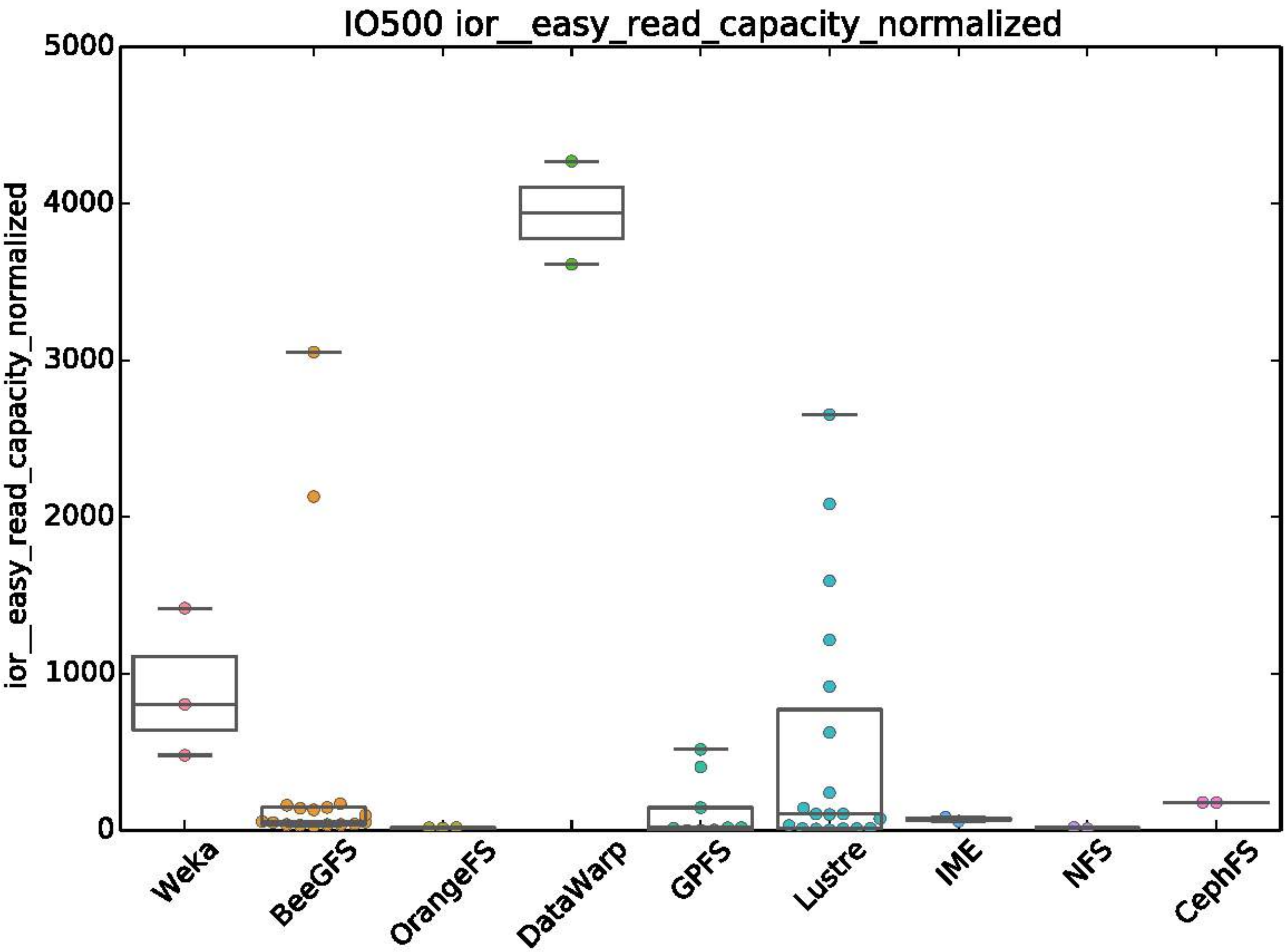
IO500


IO500 io500__score_capacity_normalized
io500__score_capacity_normalized
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

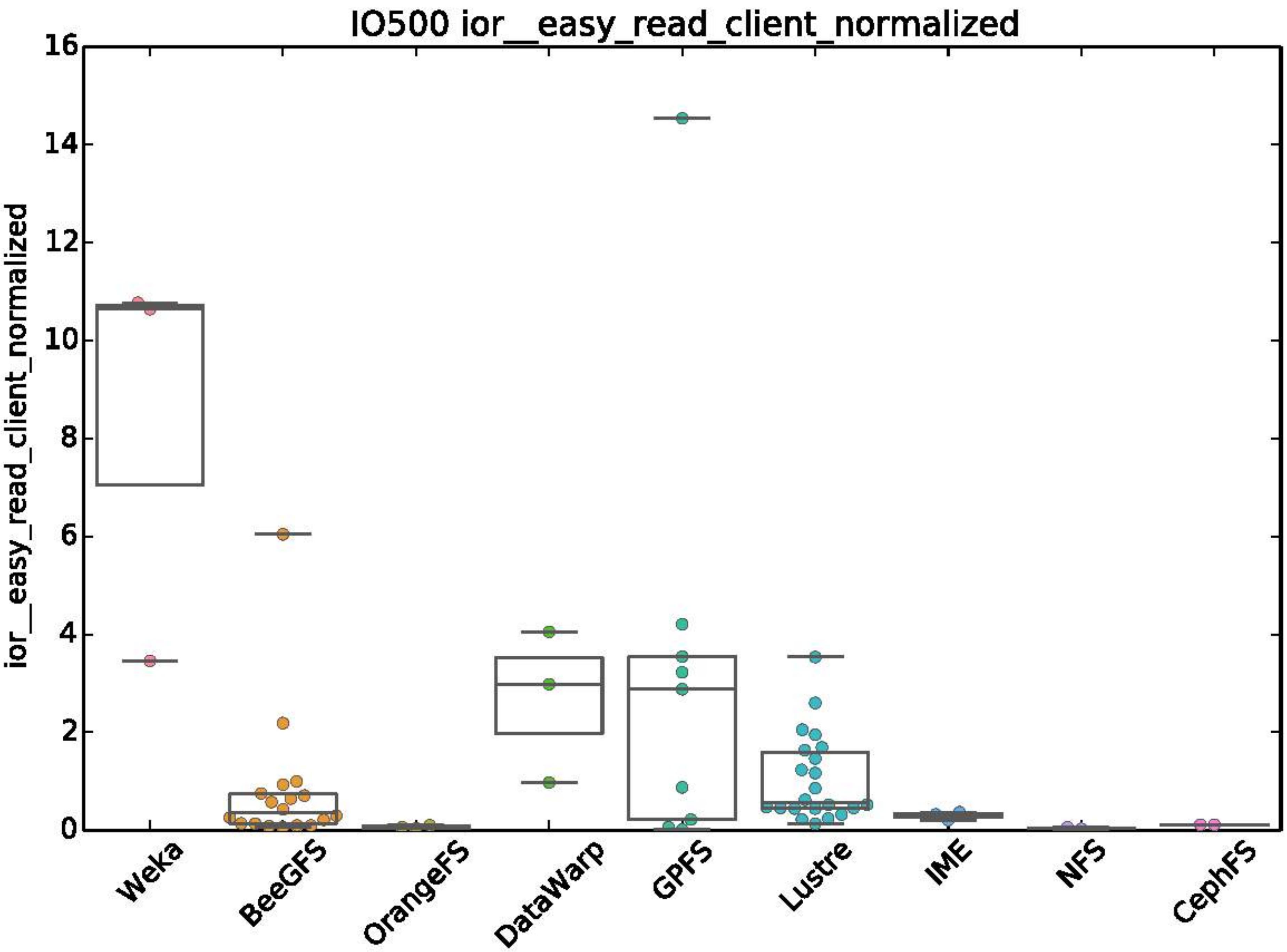

IO500 io500__score_client_normalized
io500__score_client_normalized
0
1
2
3
4
5
6
7
8
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

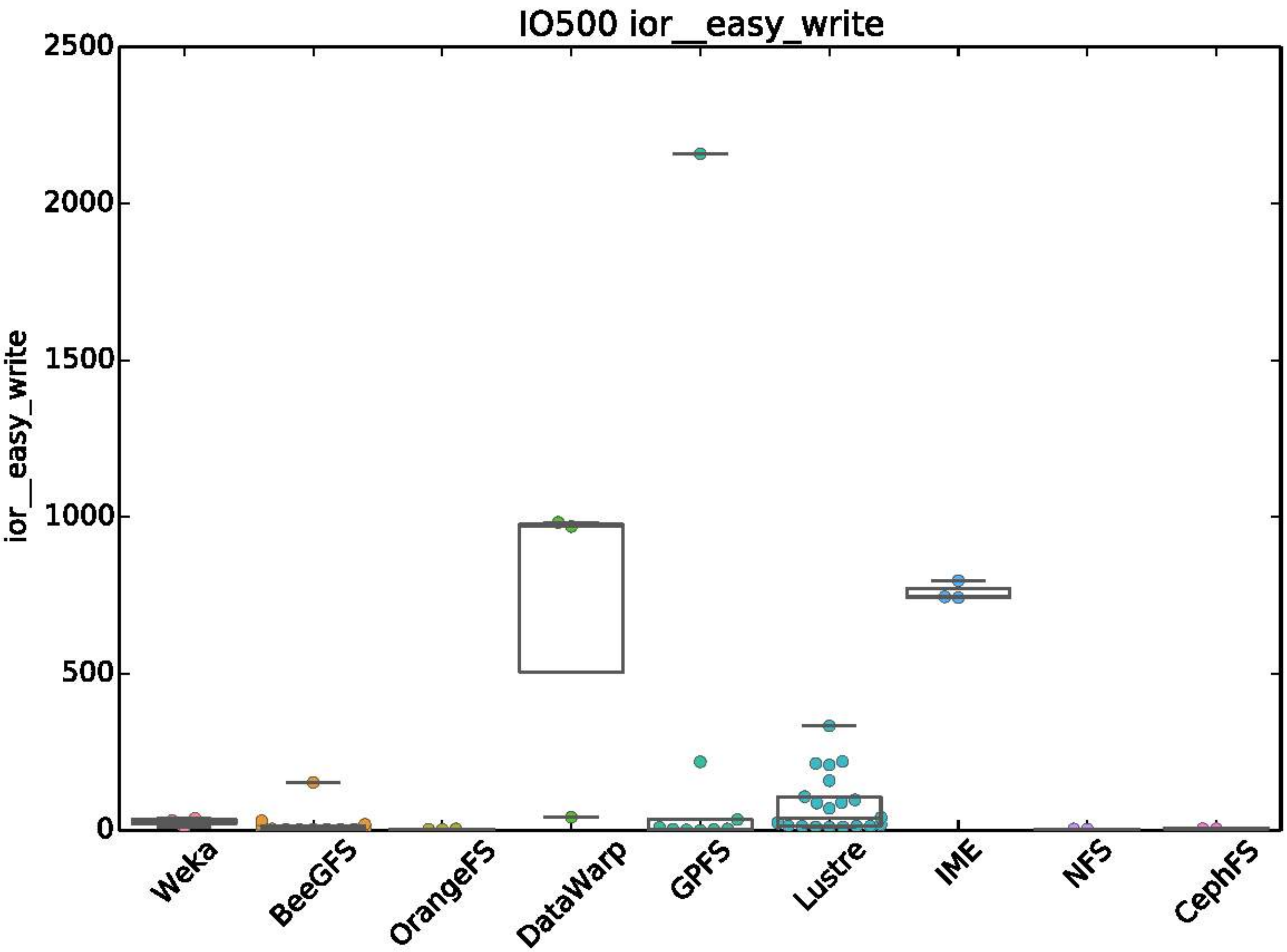
IO500


IO500 ior__easy_read
ior__easy_read
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS


IO500 ior__easy_read_capacity_normalized
ior__easy_read_capacity_normalized
5000
4000
3000
2000
1000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

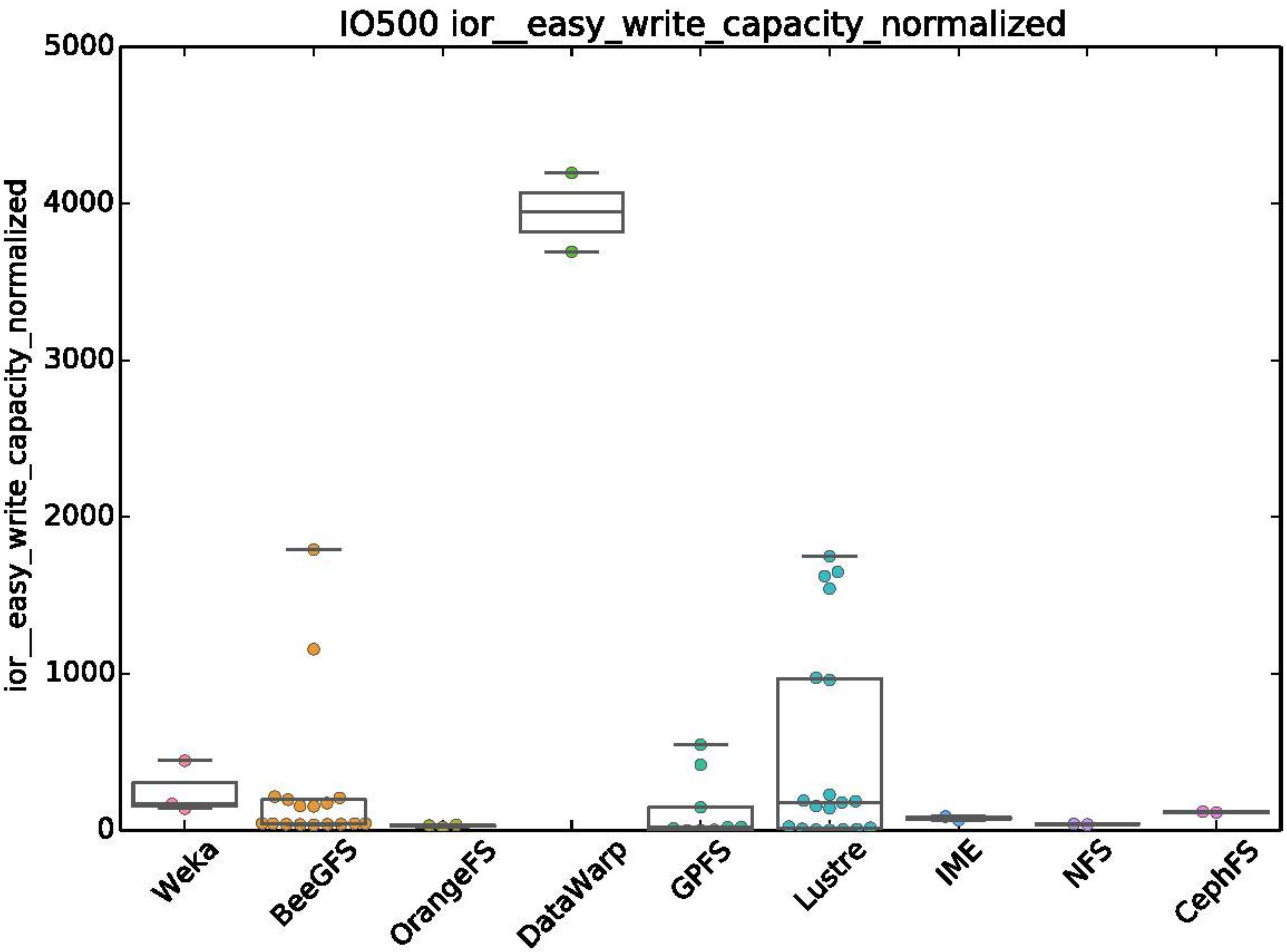
IO500

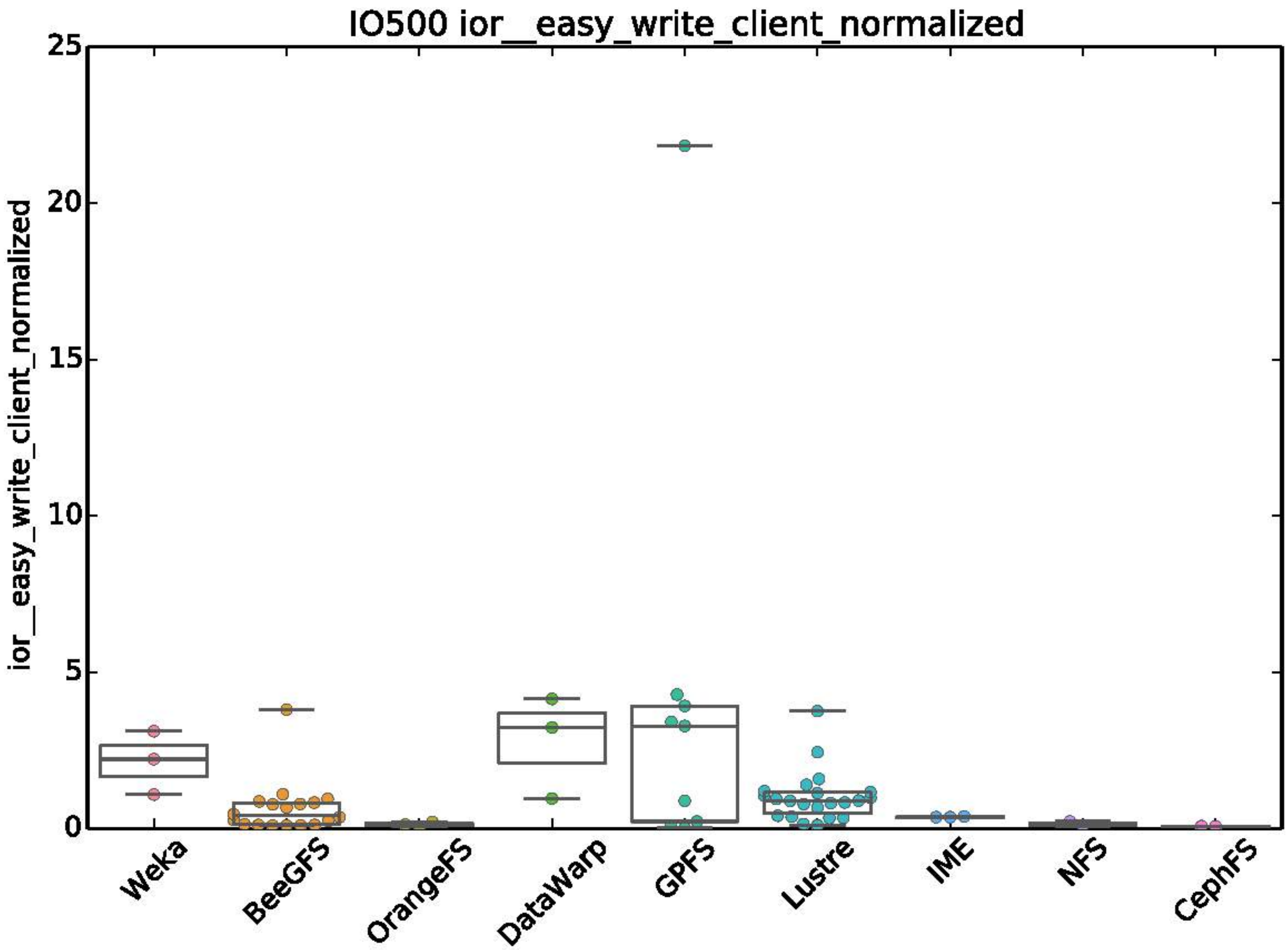

IO500 ior__easy_read_client_normalized
ior__easy_read_client_normalized
0
2
4
6
8
10
12
14
16
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS



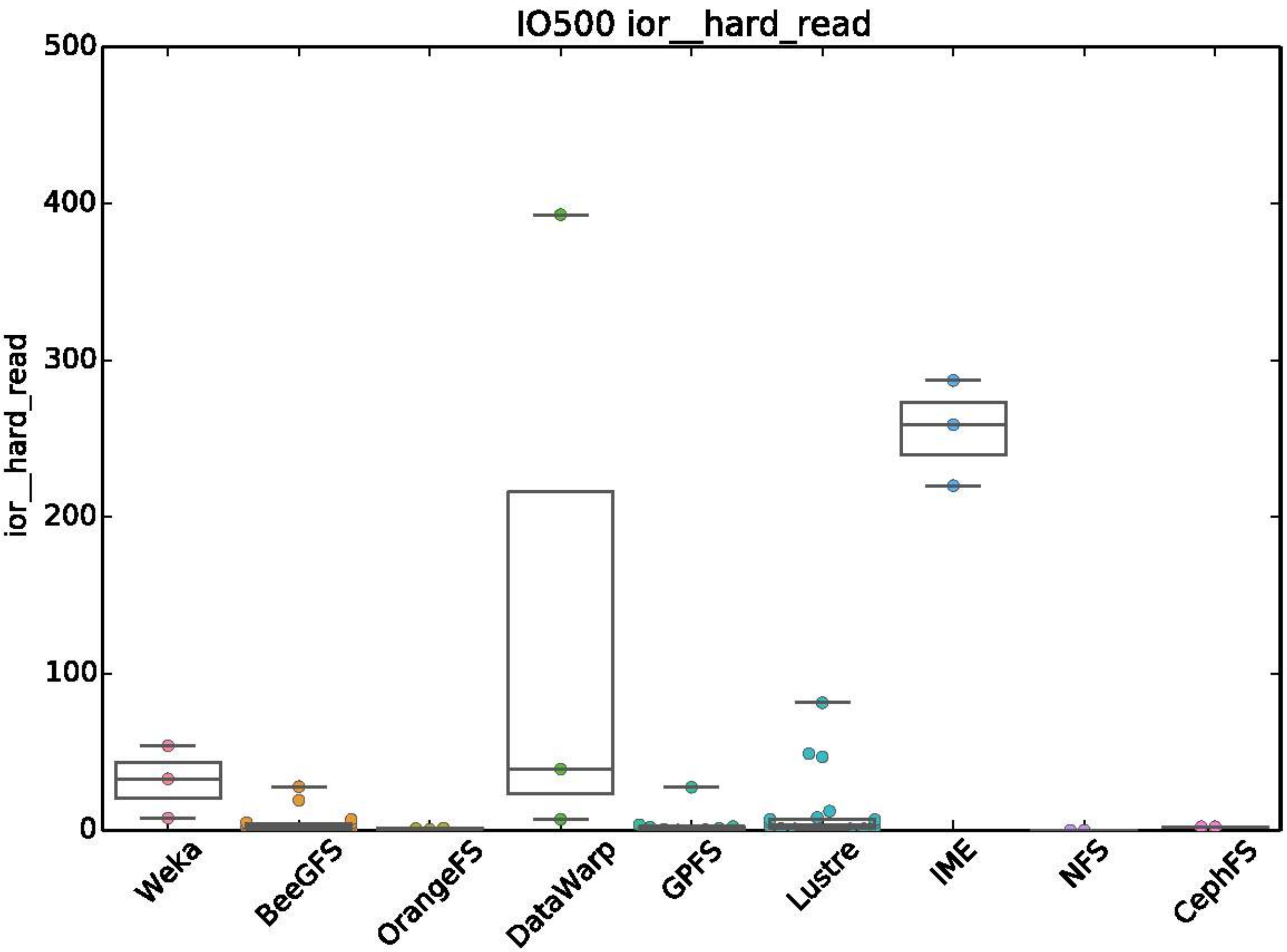
IO500 ior__easy_write
ior__easy_write
0
500
1000
1500
2000
2500
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

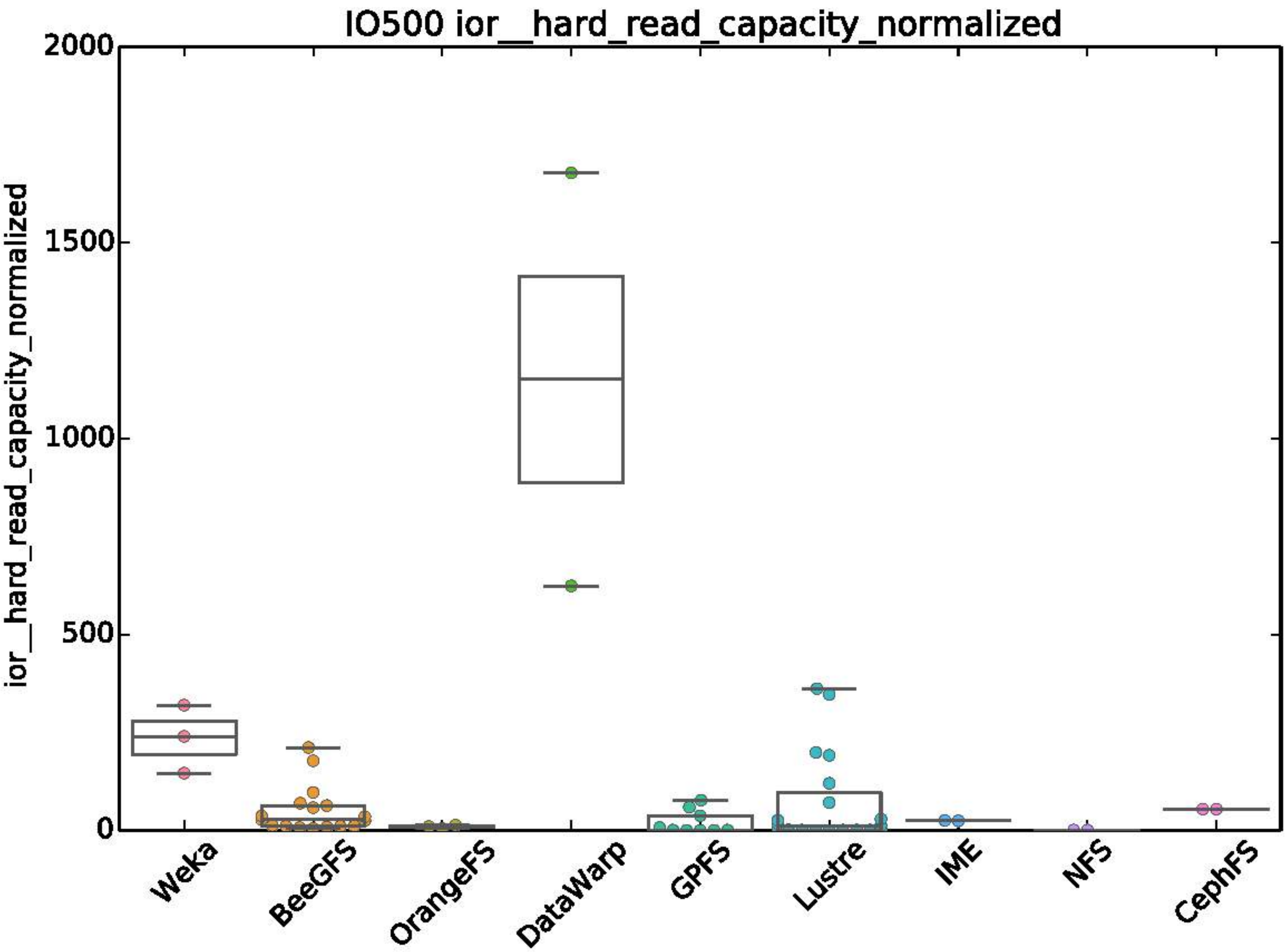
IO500

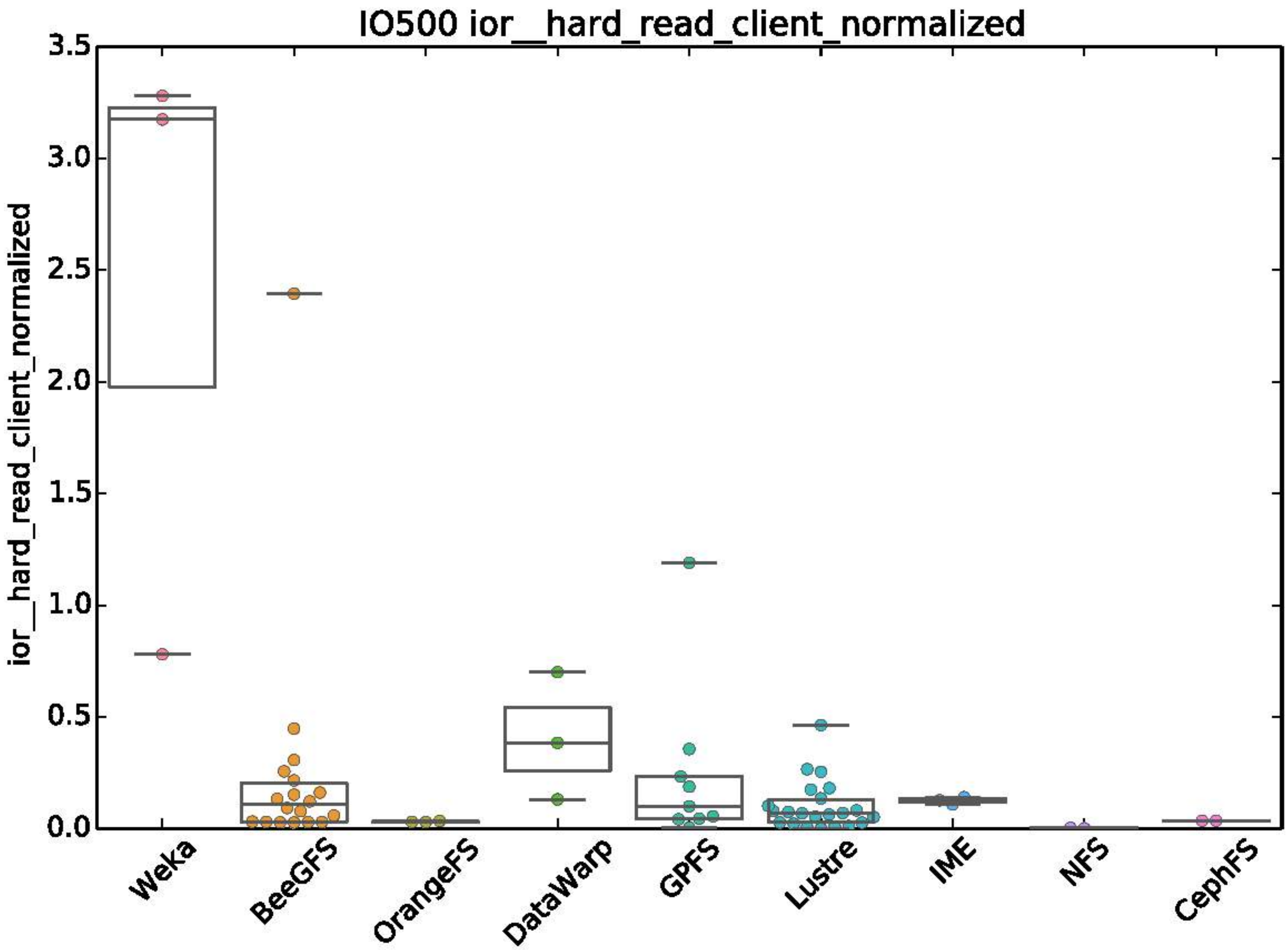

IO500 ior__easy_write_capacity_normalized
ior__easy_write_capacity_normalized
5000
4000
3000
2000
1000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS



IO500 ior__easy_write_client_normalized
ior__easy_write_client_normalized
0
5
10
15
20
25
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

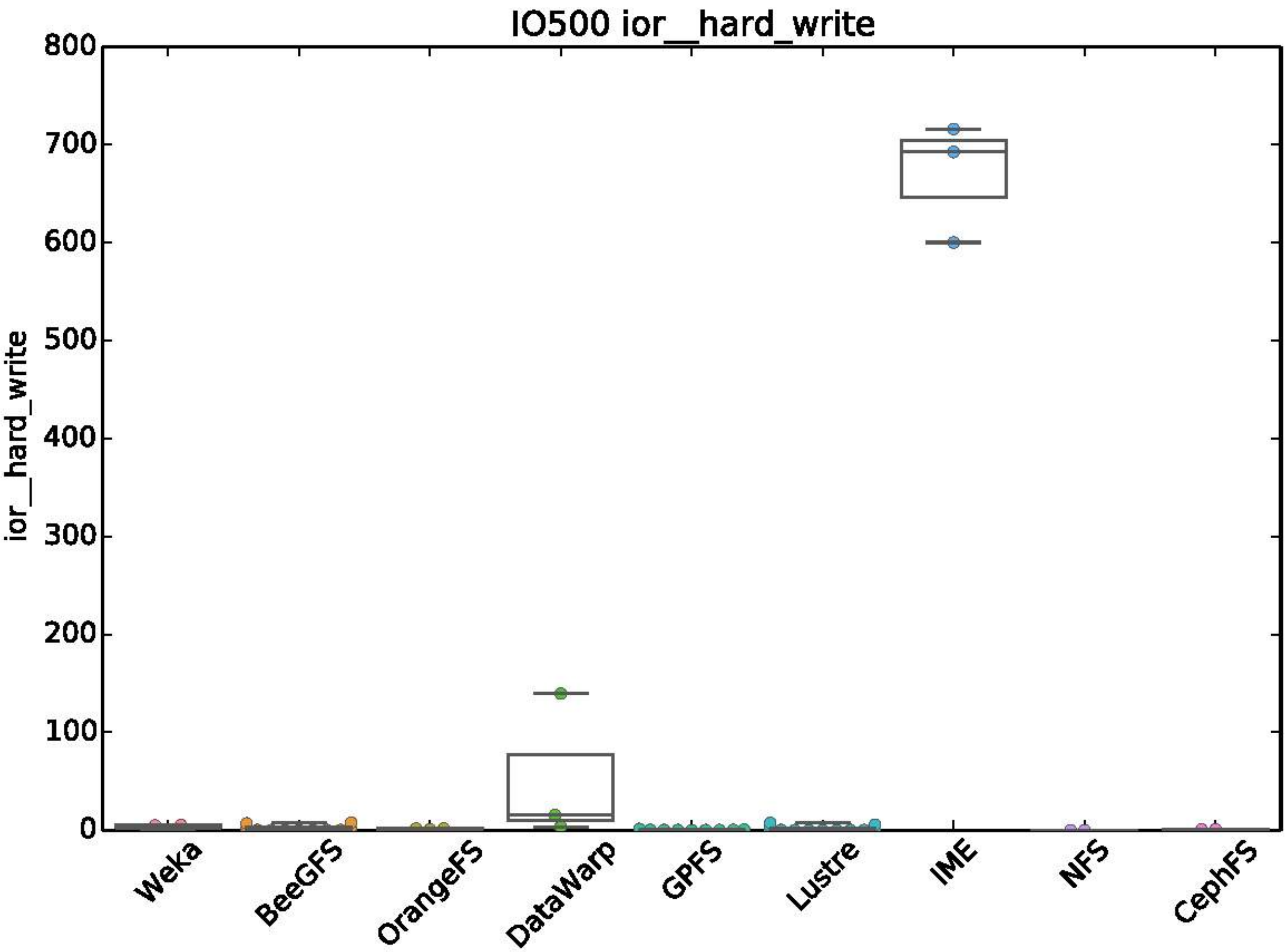

IO500 ior__hard_read
ior__hard_read
500
400
300
200
100
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

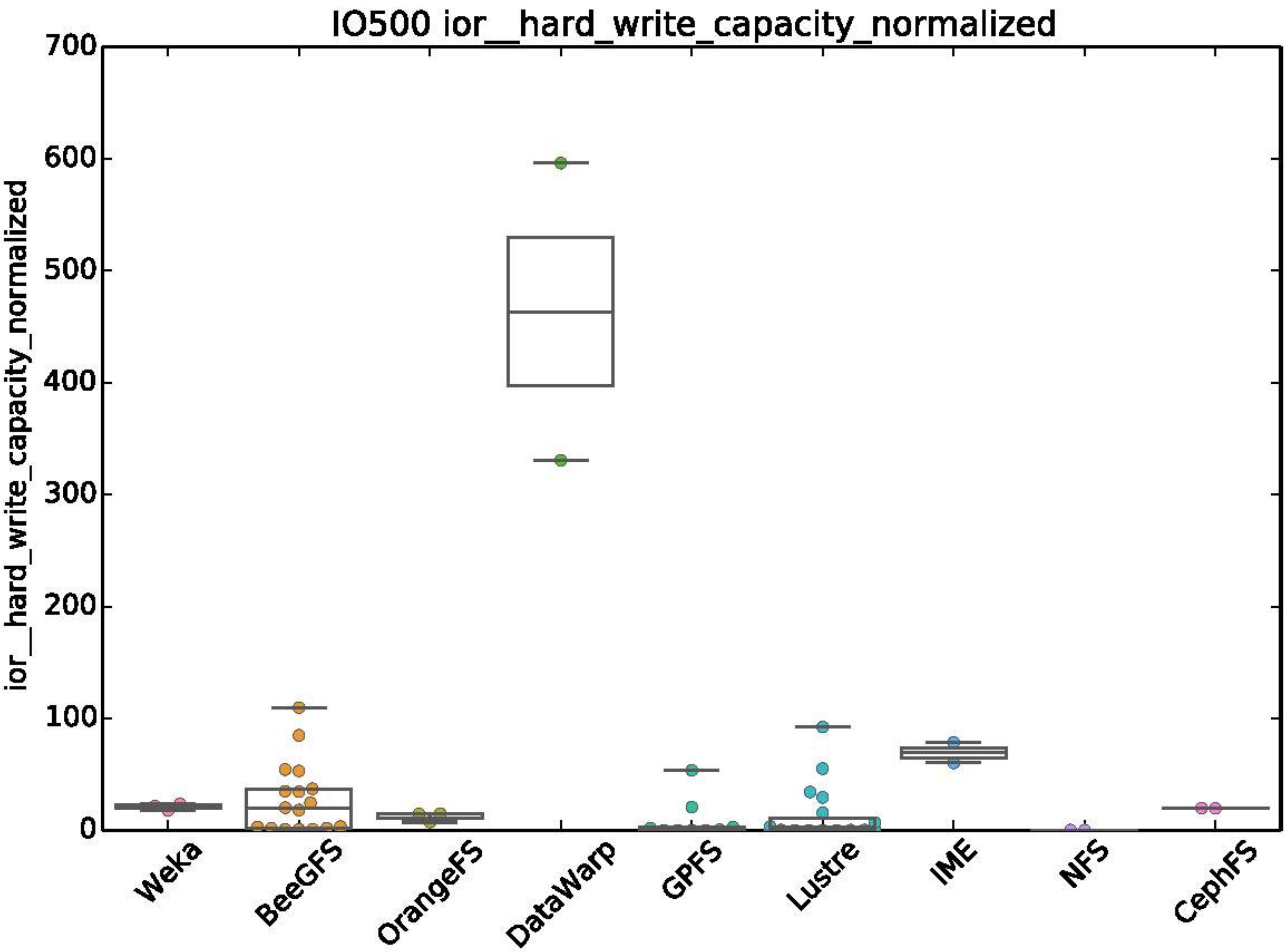

IO500 ior__hard_read_capacity_normalized
ior__hard_read_capacity_normalized
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS



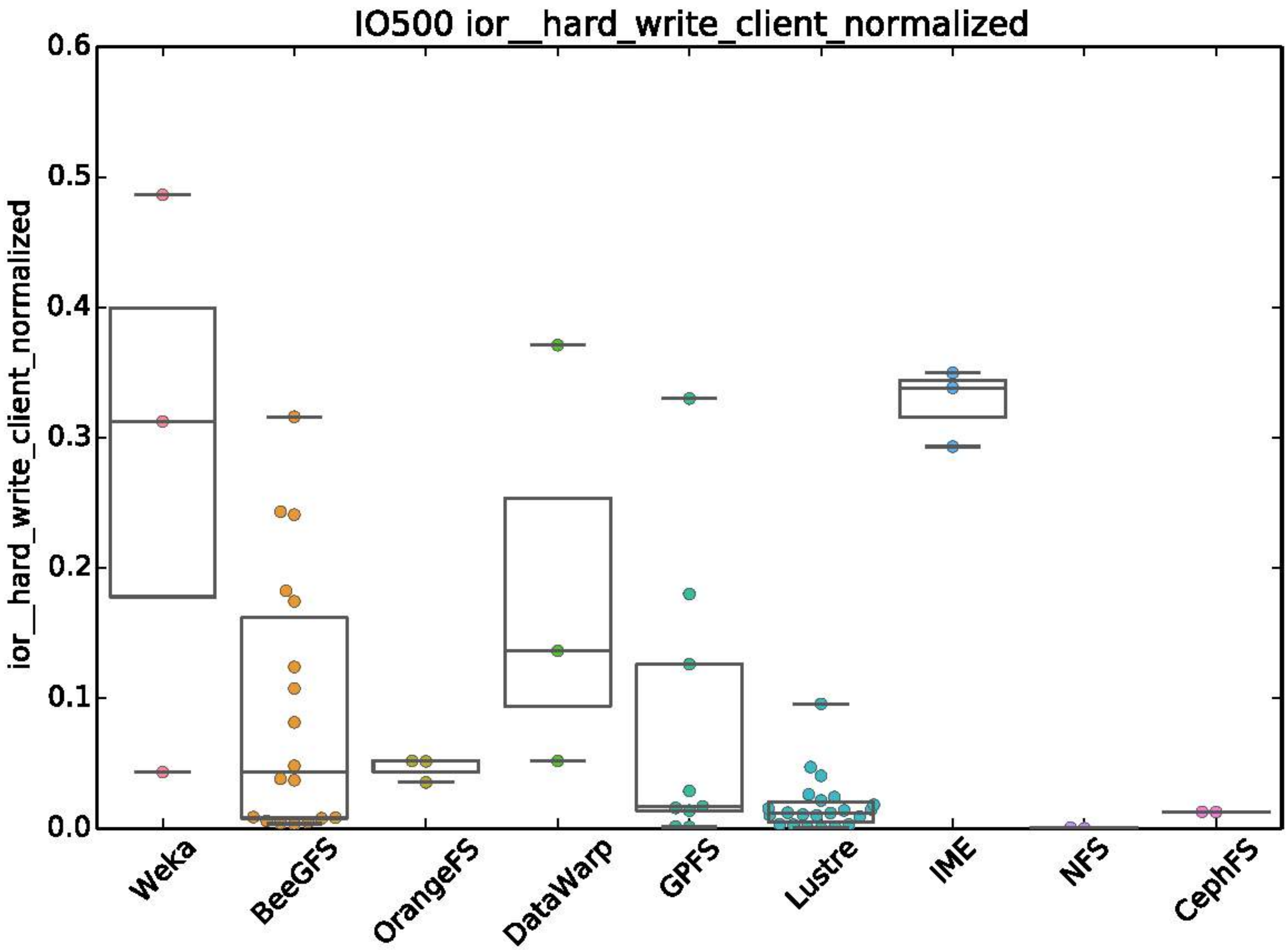
IO500 ior__hard_read_client_normalized
ior__hard_read_client_normalized
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

IO500

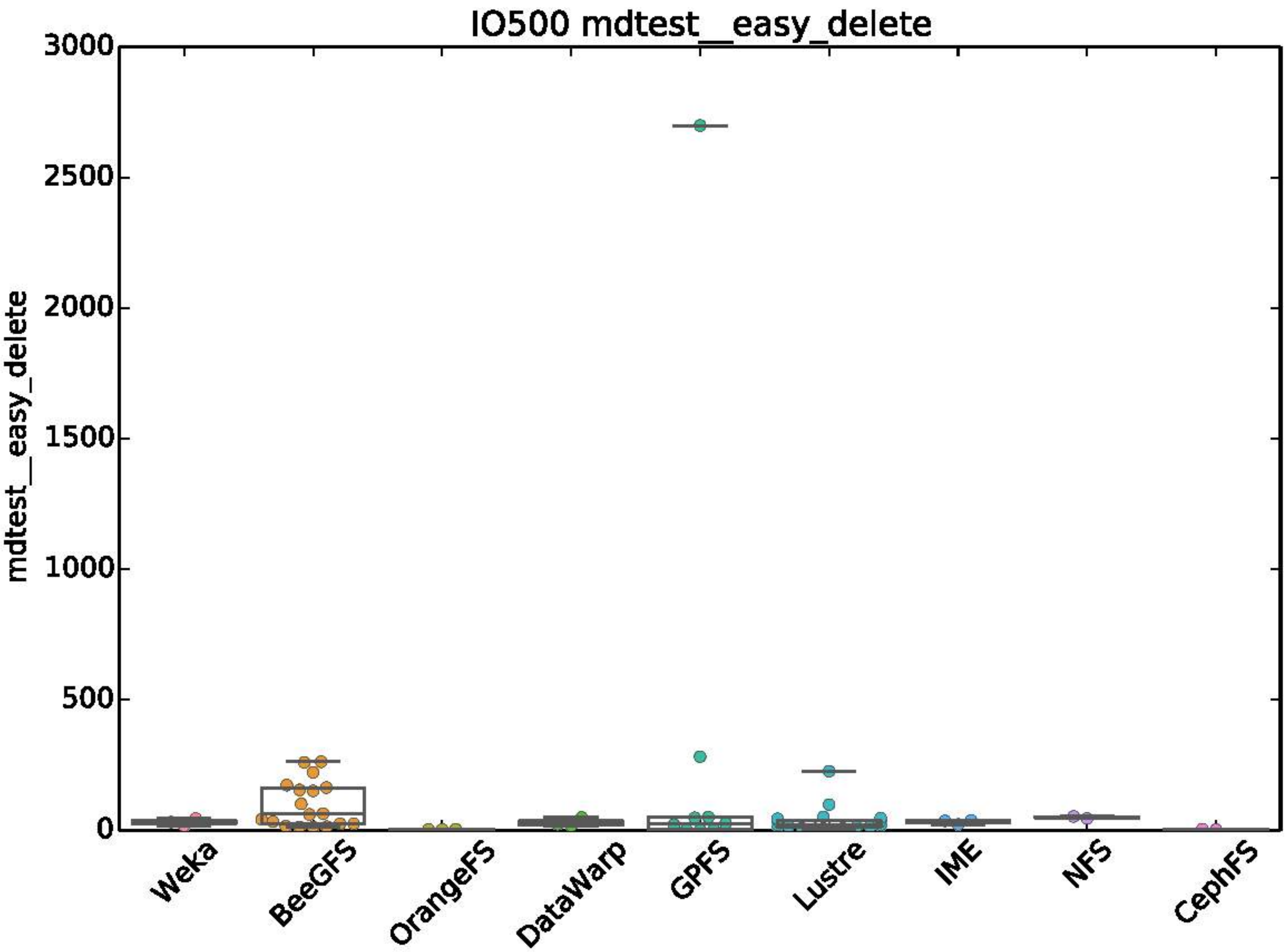

IO500 ior__hard_write
ior__hard_write
800
700
600
500
400
300
200
100
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

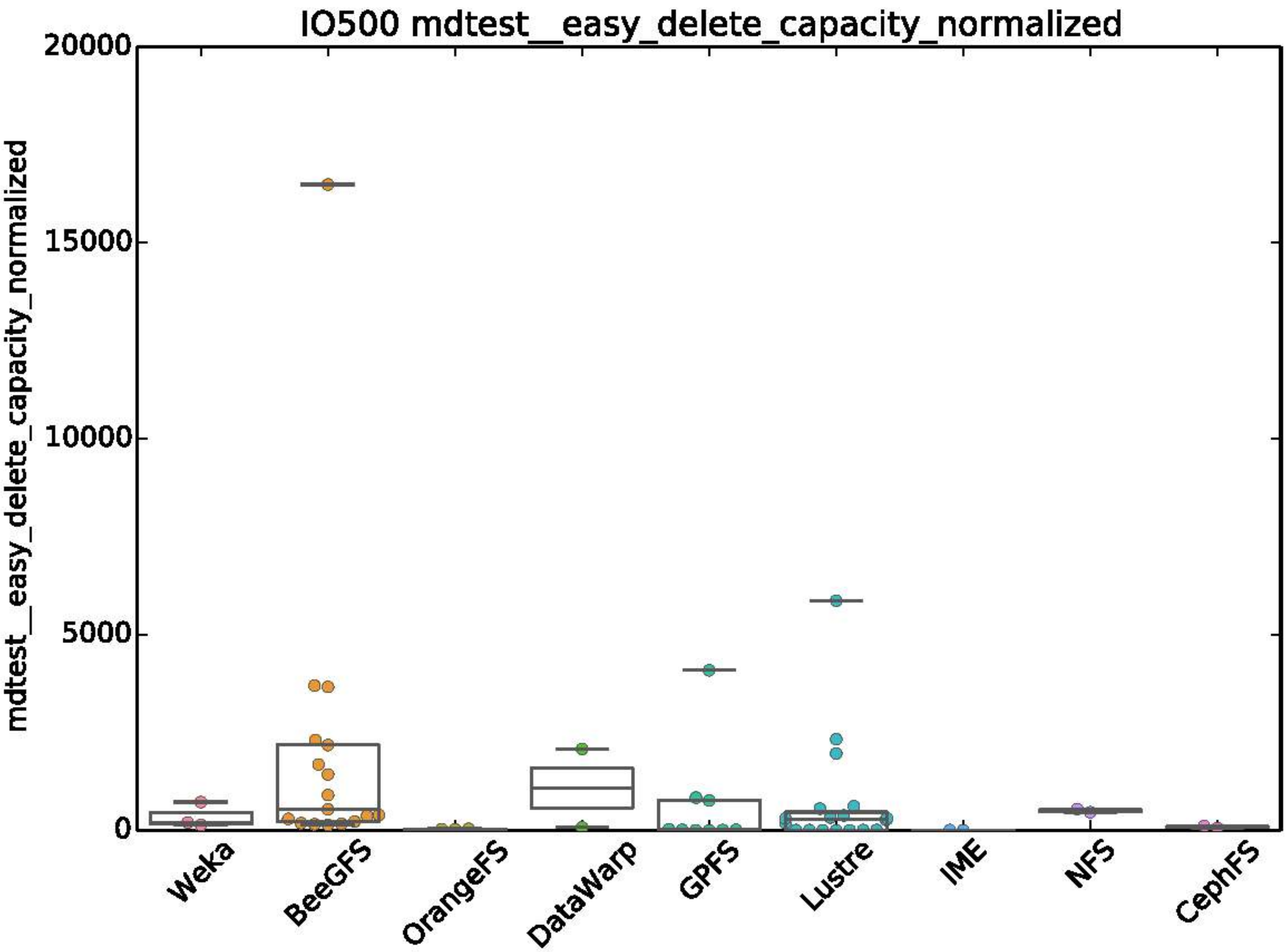


IO500 ior__hard_write_capacity_normalized
ior__hard_write_capacity_normalized
700
600
500
400
300
200
100
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS



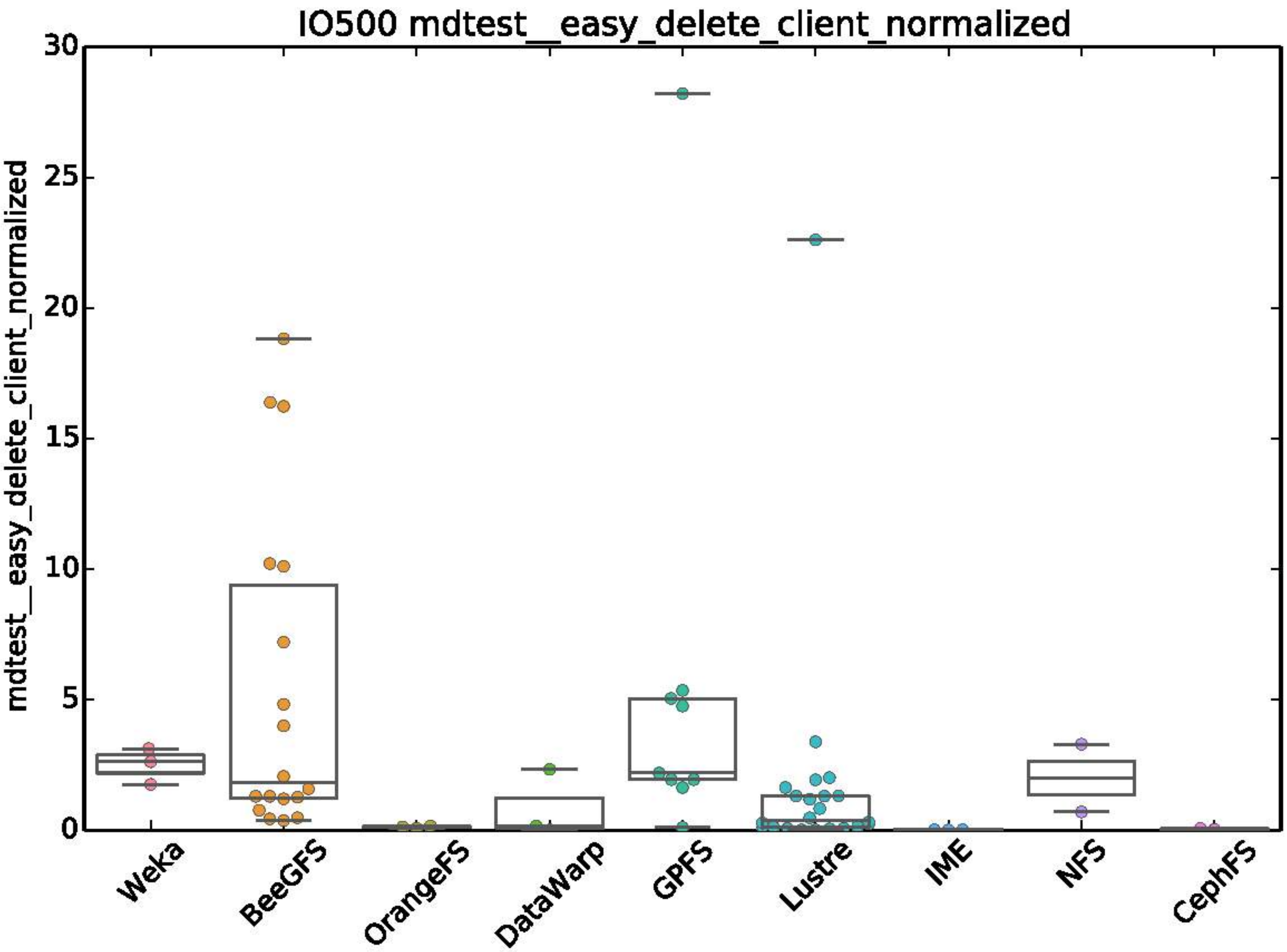
IO500@SC18; Bent

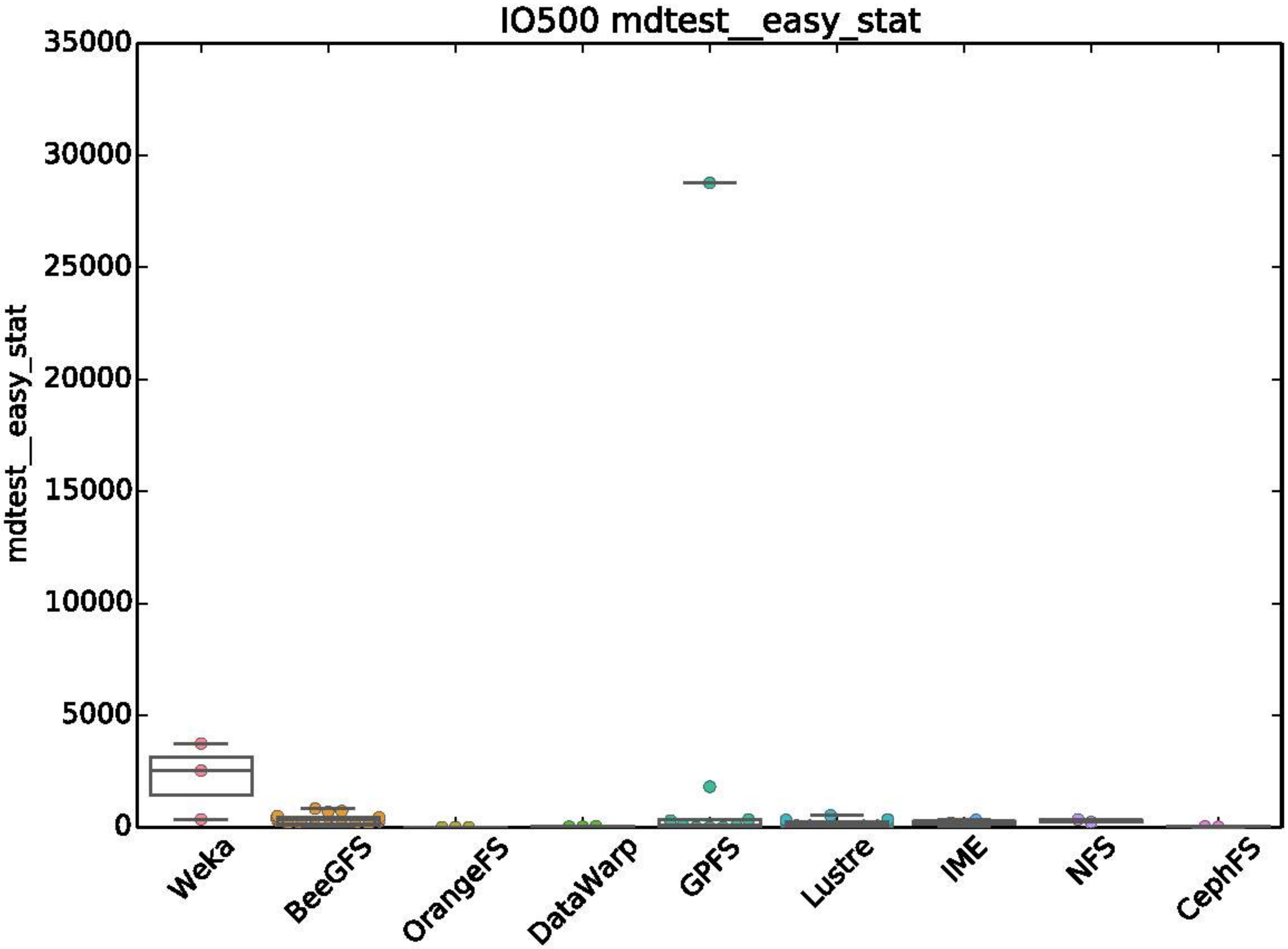

IO500 mdtest__easy_delete
mdtest__easy_delete
3000
2500
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS



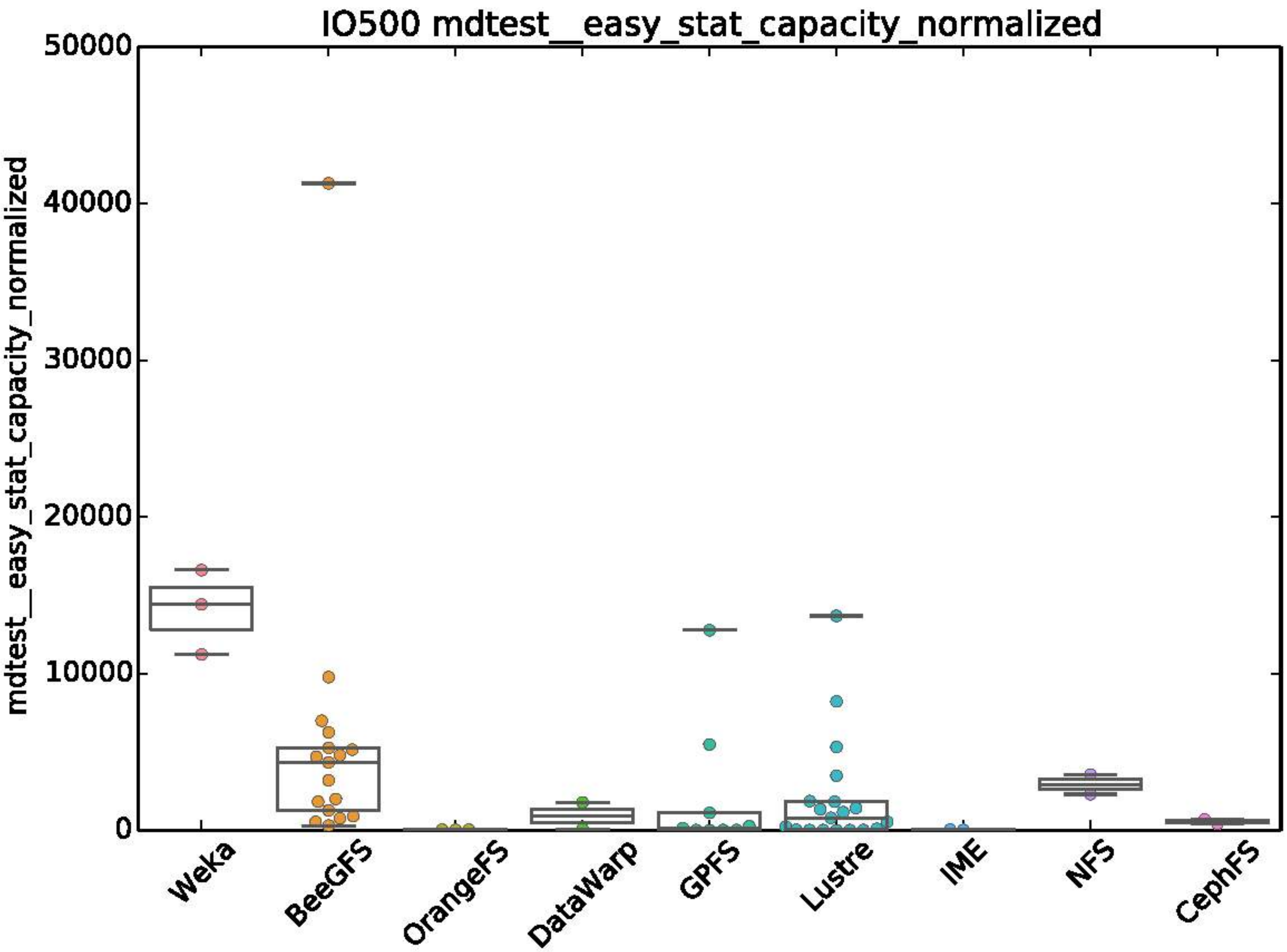
IO500 mdtest__easy_delete_capacity_normalized
mdtest__easy_delete_capacity_normalized
20000
15000
10000
5000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

IO500

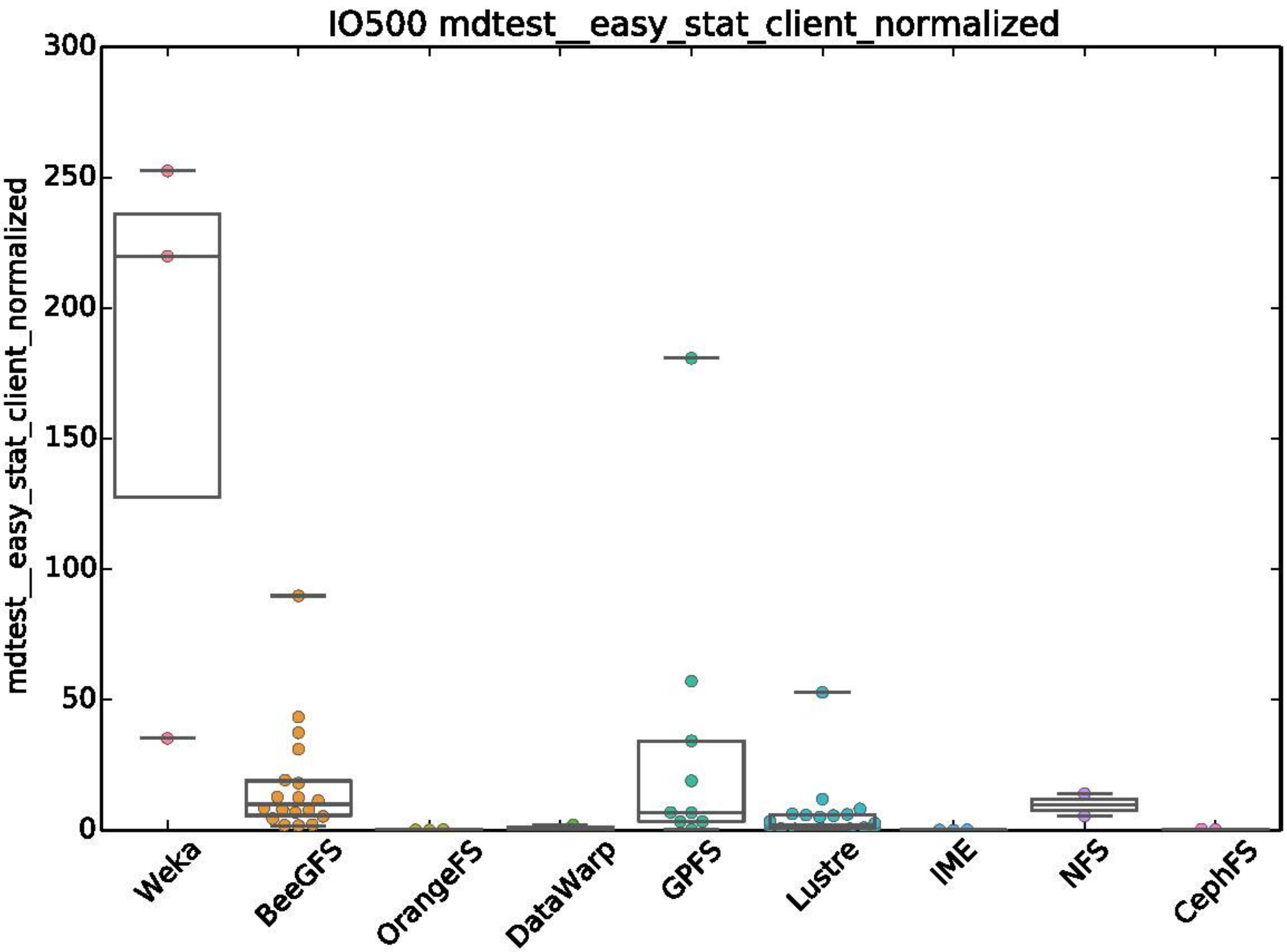

IO500 mdtest__easy_delete_client_normalized
mdtest__easy_delete_client_normalized
0
5
10
15
20
25
30
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS


IO500 mdtest__easy_stat
mdtest__easy_stat
35000
30000
25000
20000
15000
10000
5000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

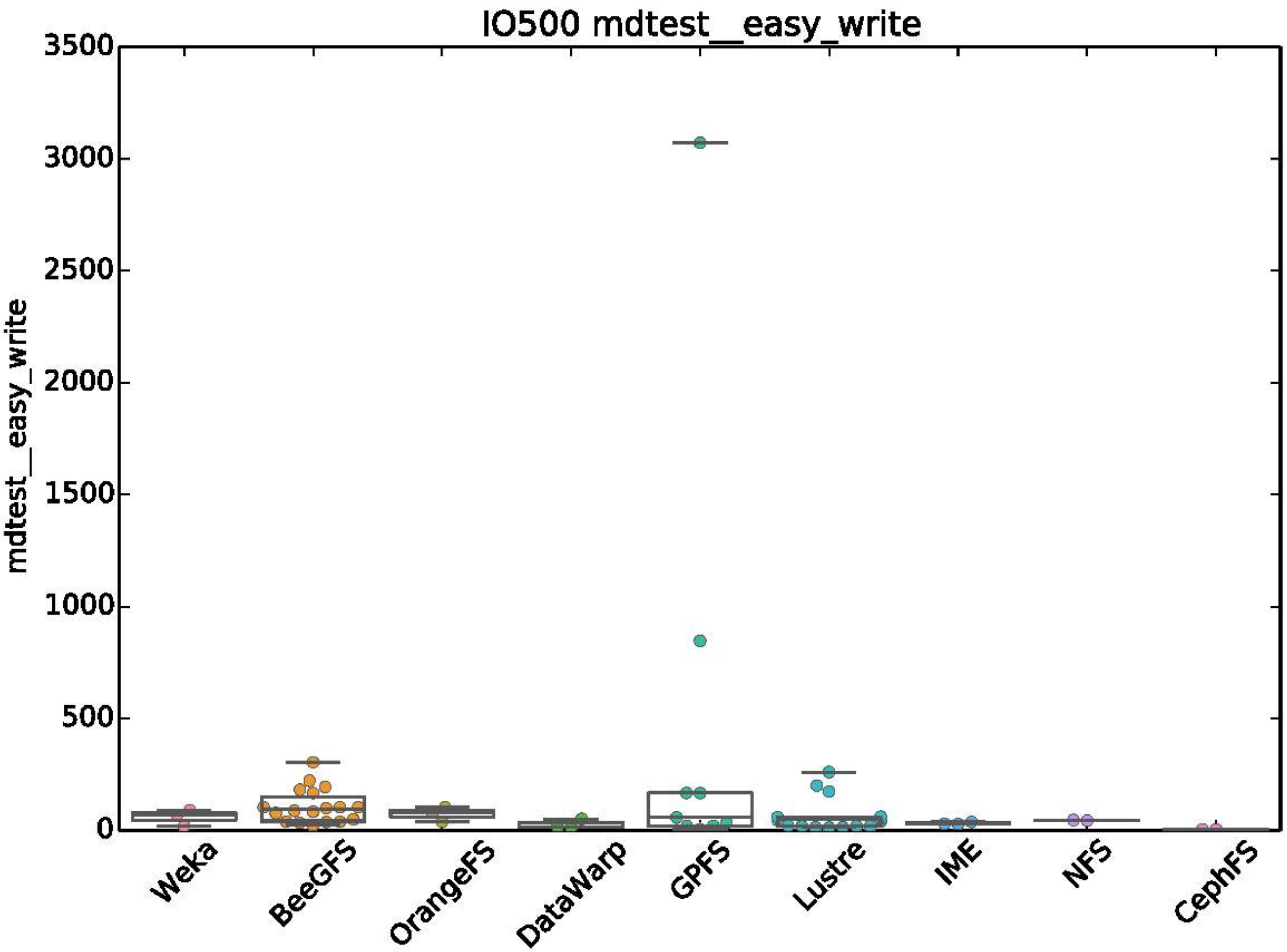
IO500


IO500 mdtest__easy_stat_capacity_normalized
mdtest__easy_stat_capacity_normalized
50000
40000
30000
20000
10000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

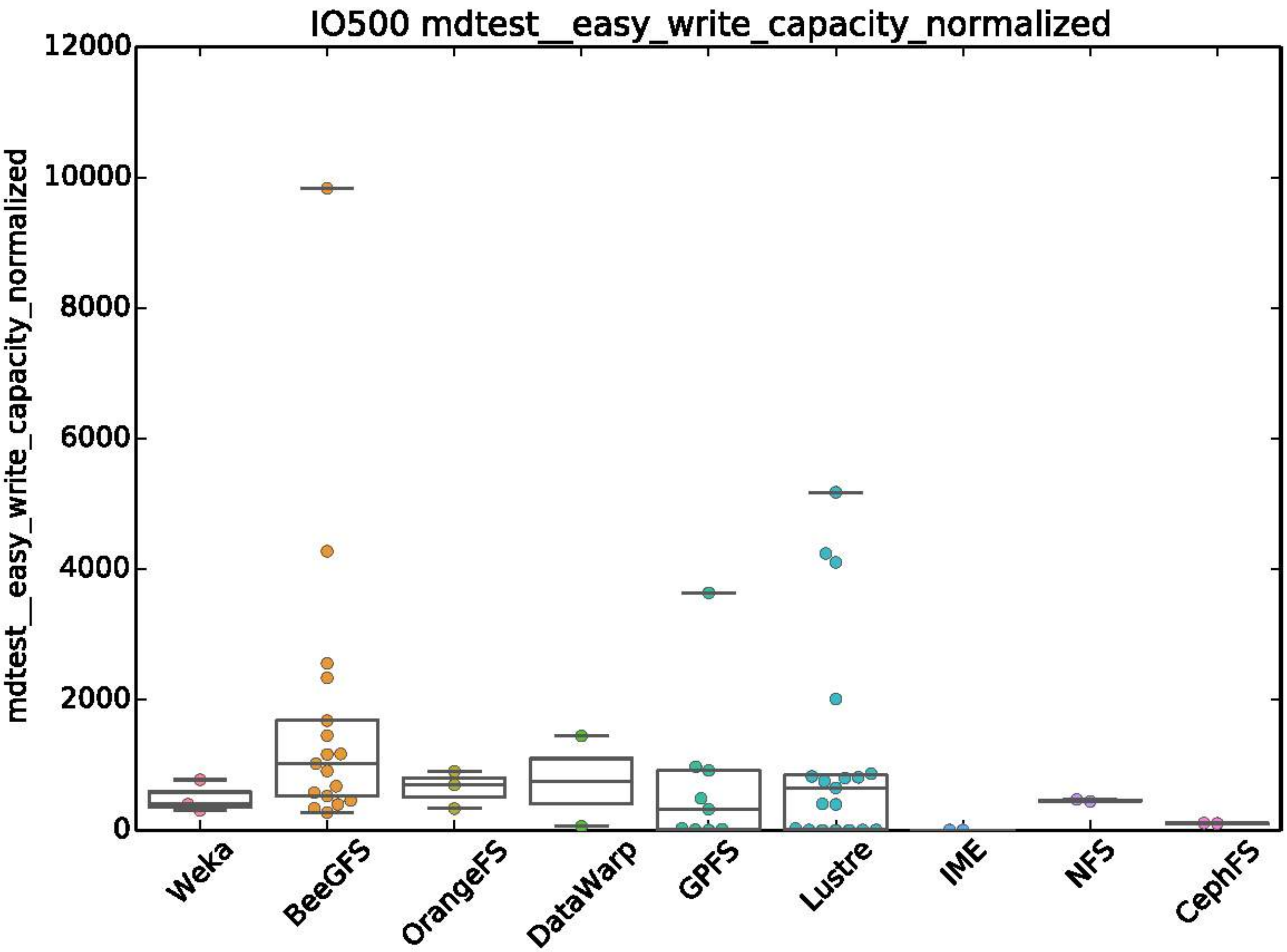
IO500


IO500 mdtest__easy_stat_client_normalized
mdtest__easy_stat_client_normalized
300
250
200
150
100
50
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

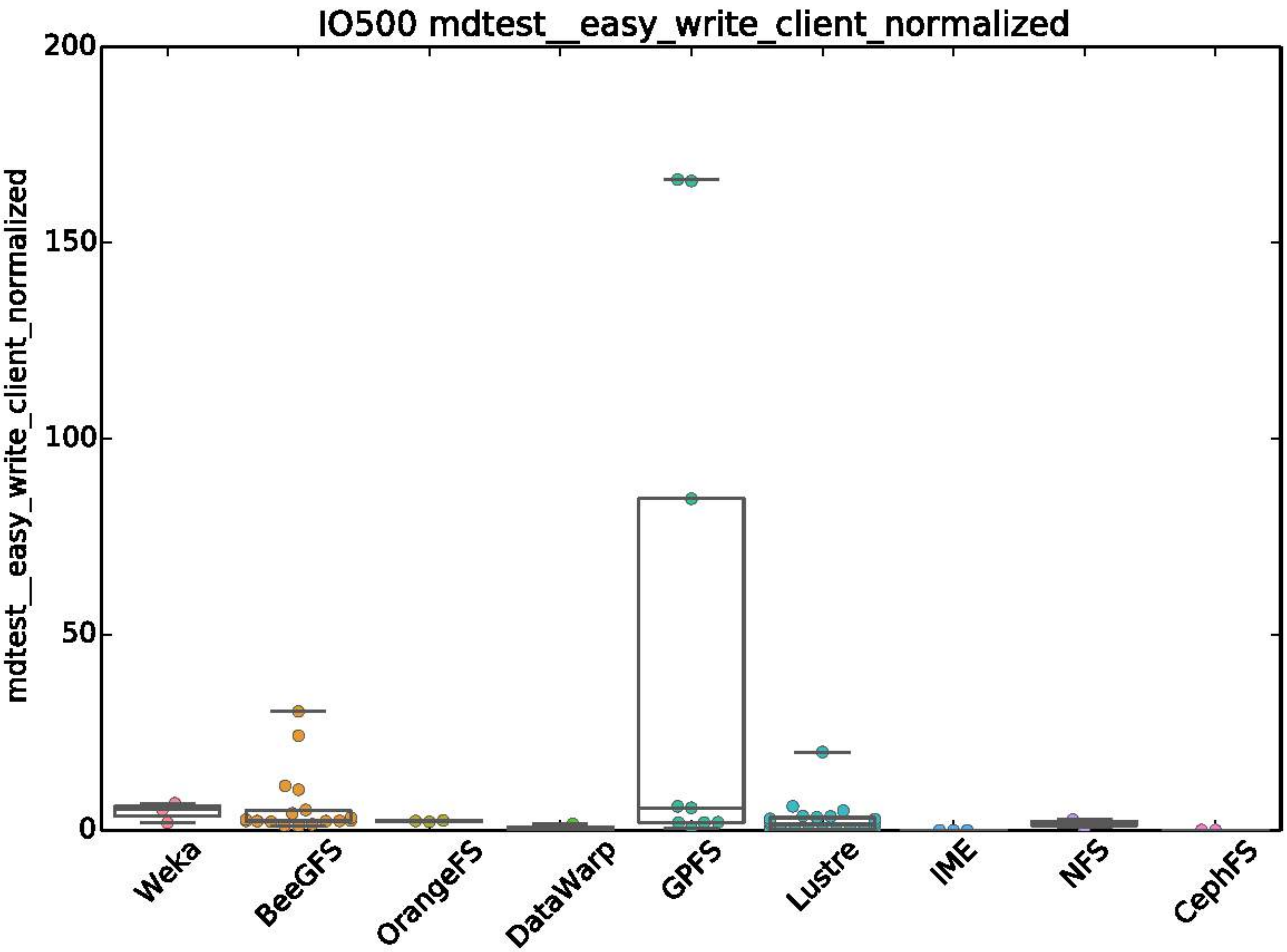
IO500



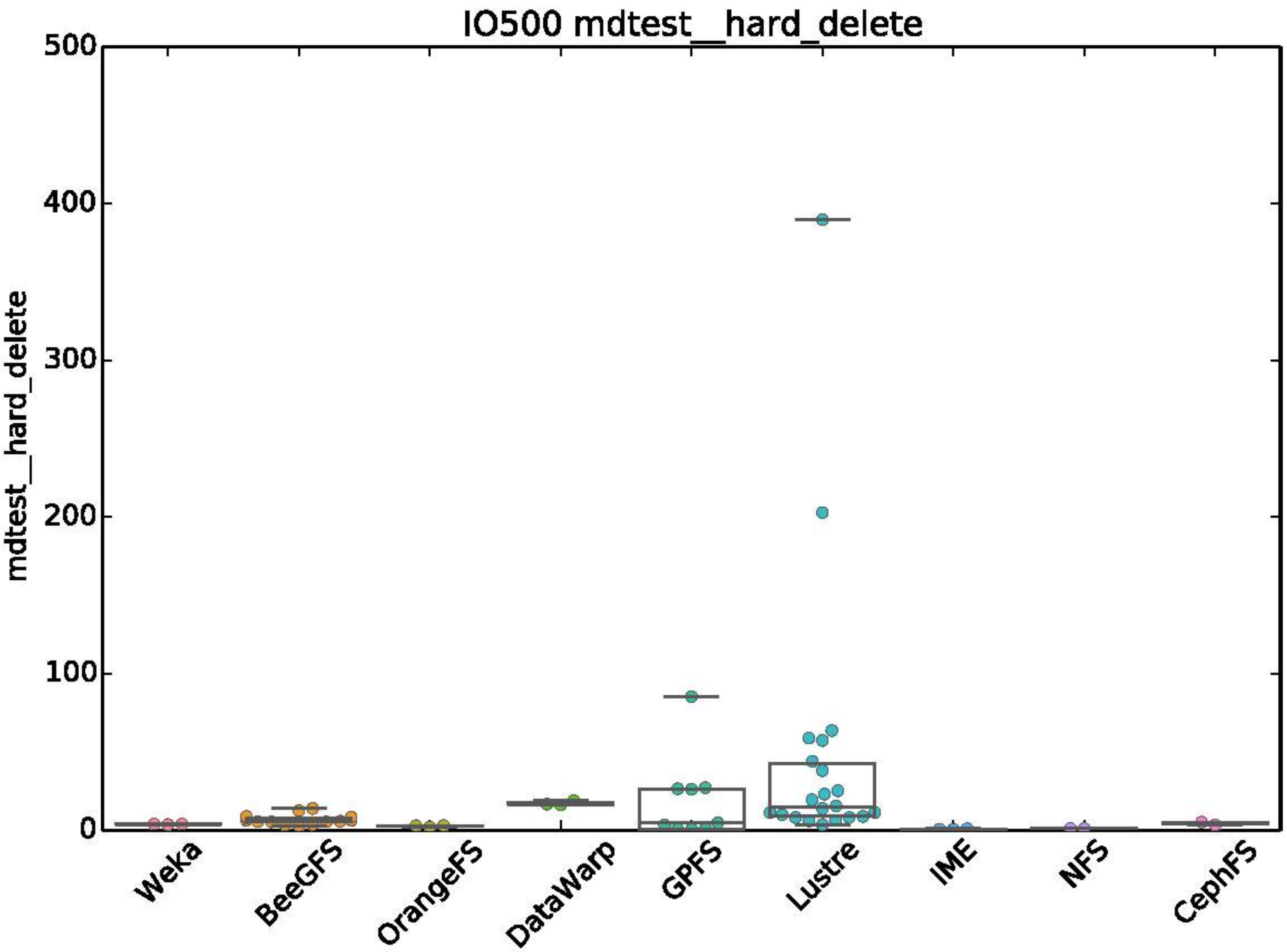
IO500 mdtest__easy_write
mdtest__easy_write
3500
3000
2500
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

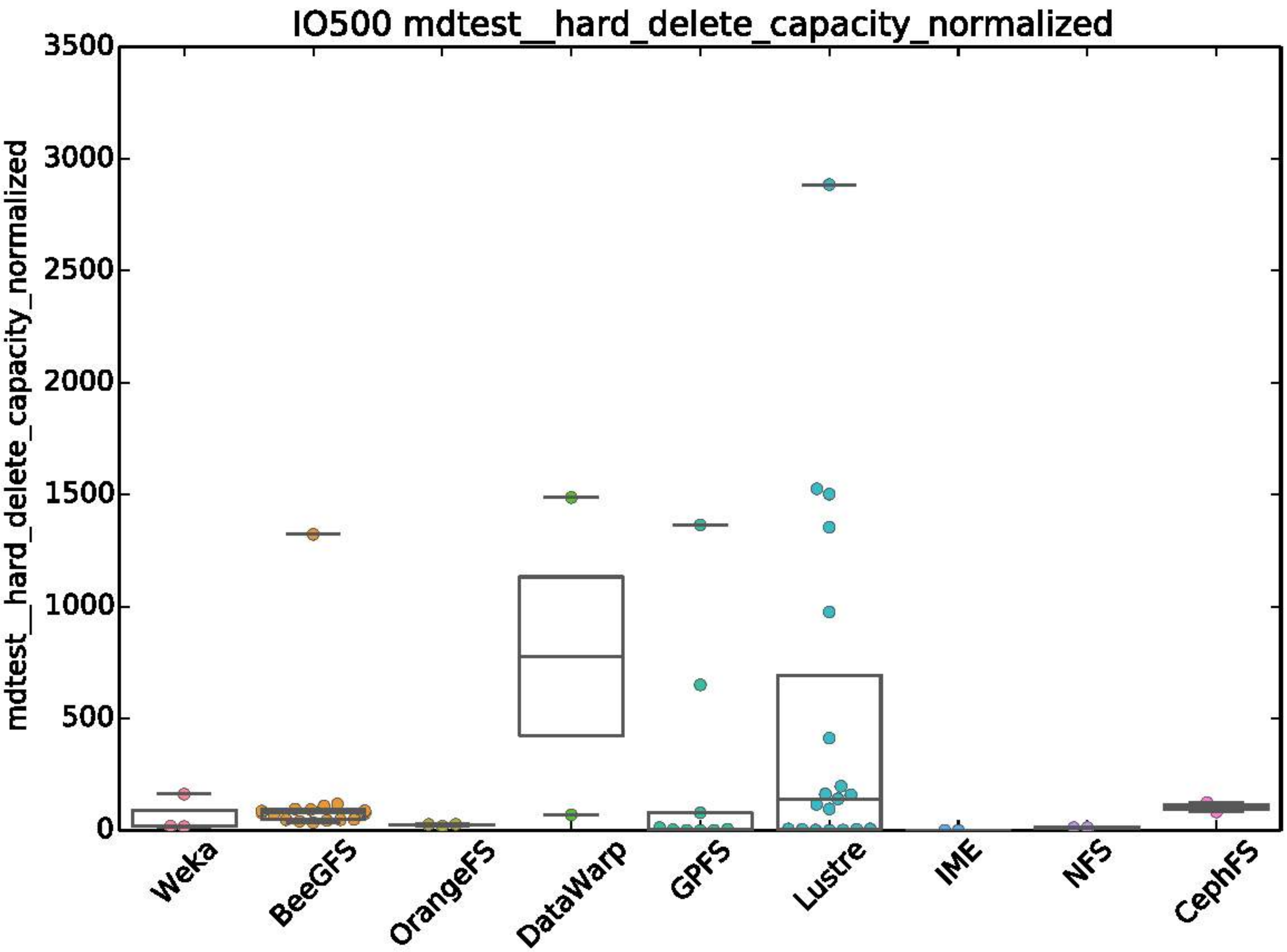

IO500 mdtest__easy_write_capacity_normalized
mdtest__easy_write_capacity_normalized
12000
10000
8000
6000
4000
2000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

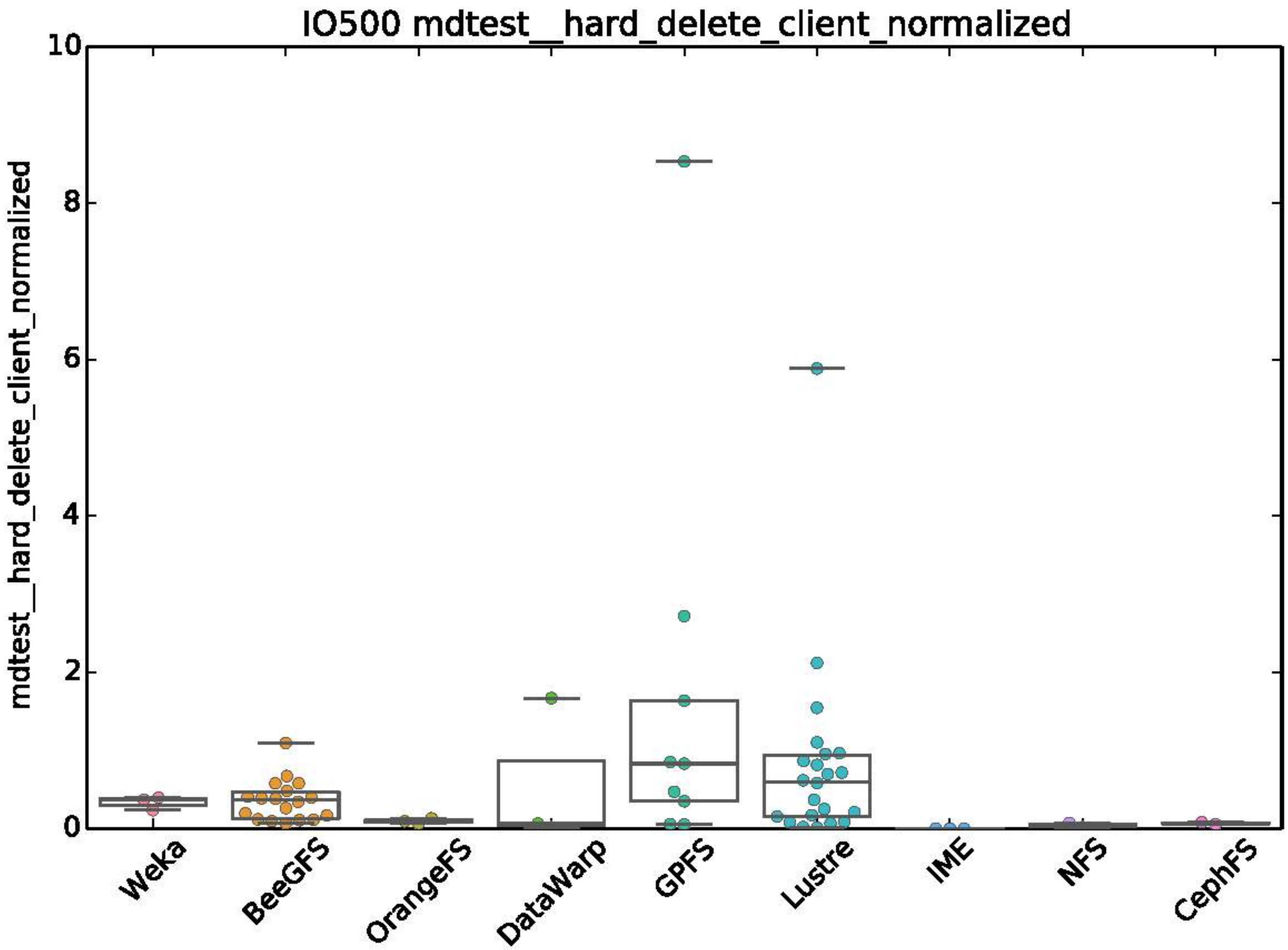
IO 500


IO500 mdtest__easy_write_client_normalized
mdtest__easy_write_client_normalized
200
150
100
50
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

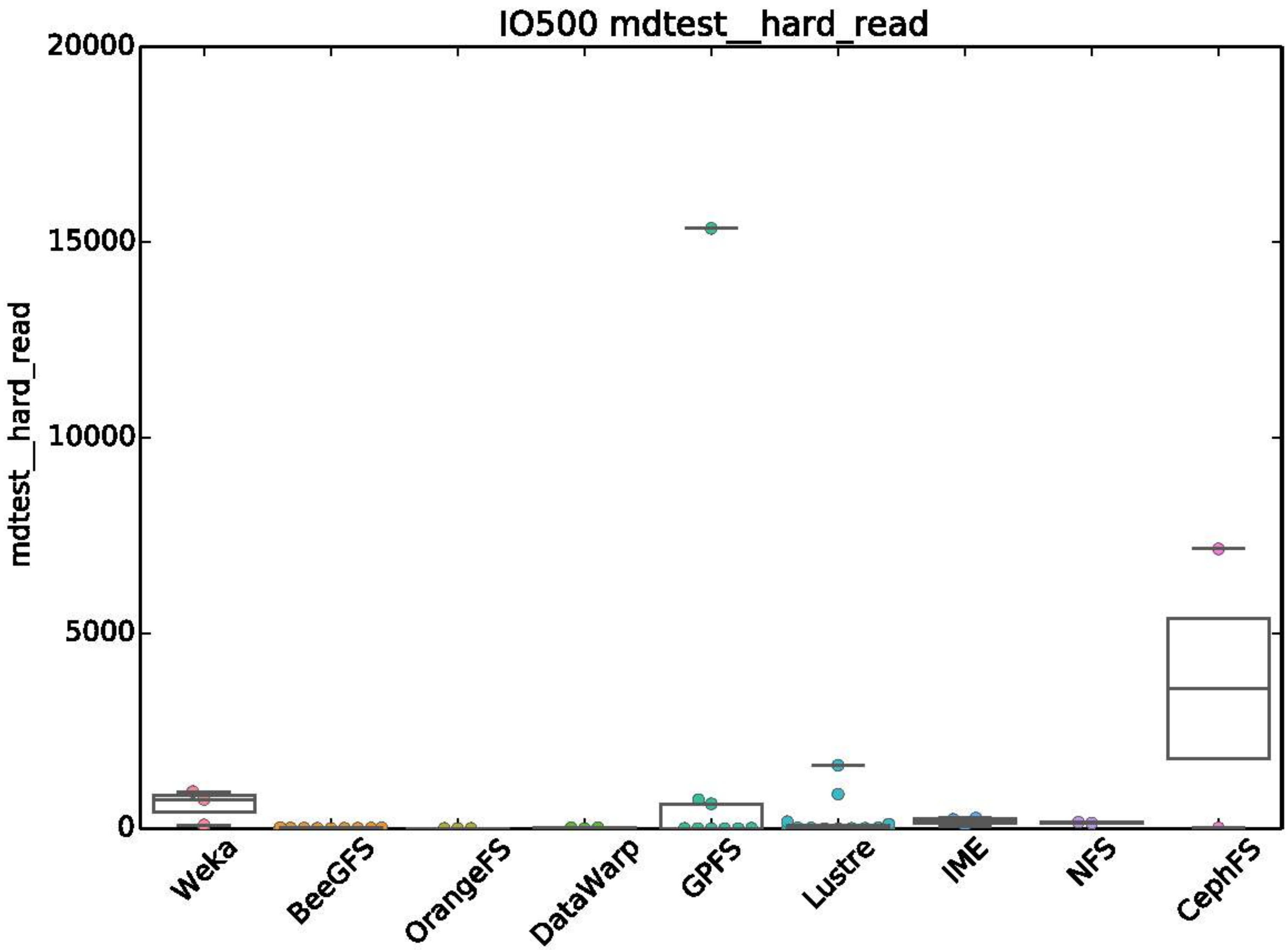
IO500

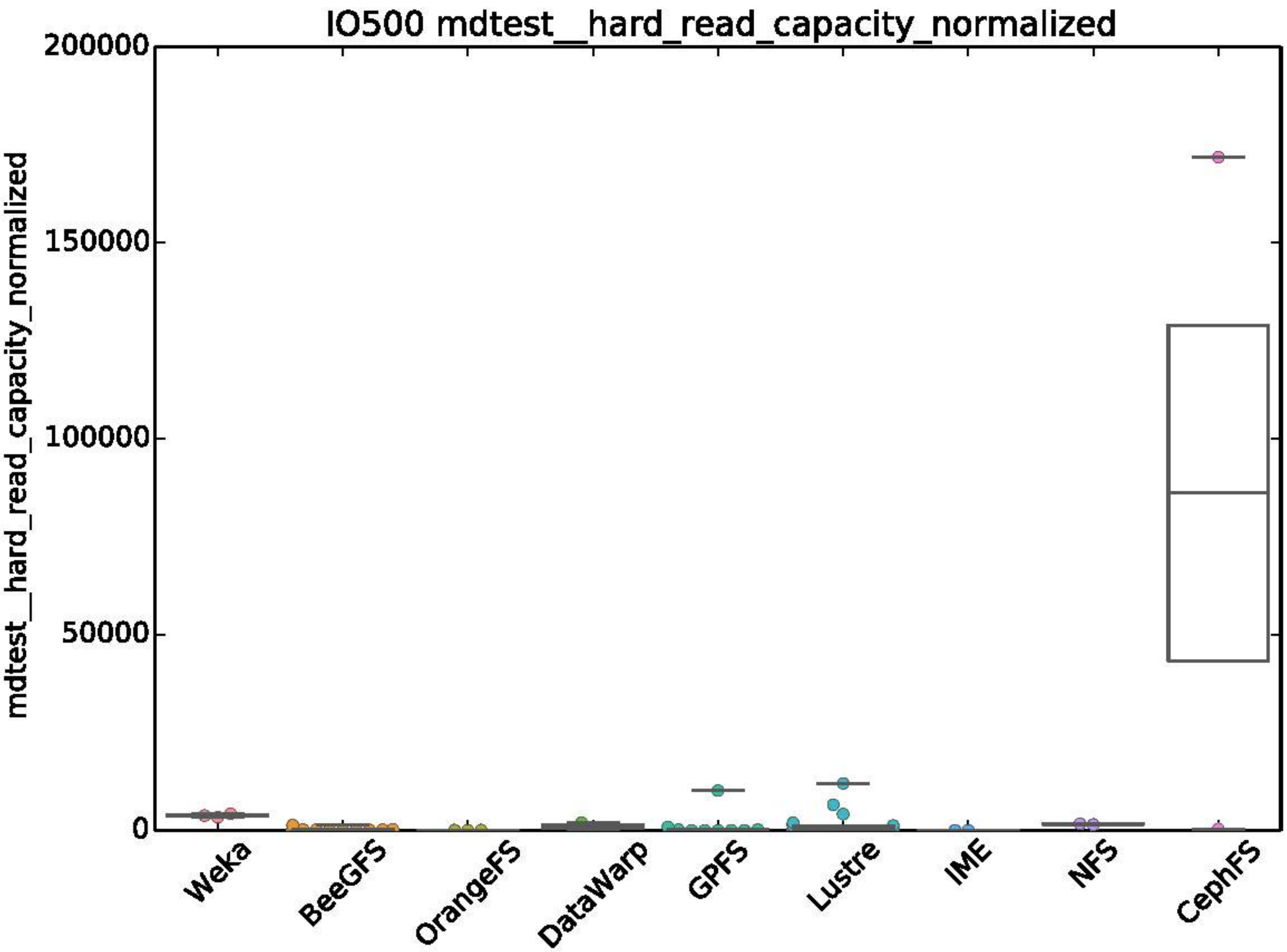

IO500 mdtest__hard_delete
mdtest__hard_delete
500
400
300
200
100
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

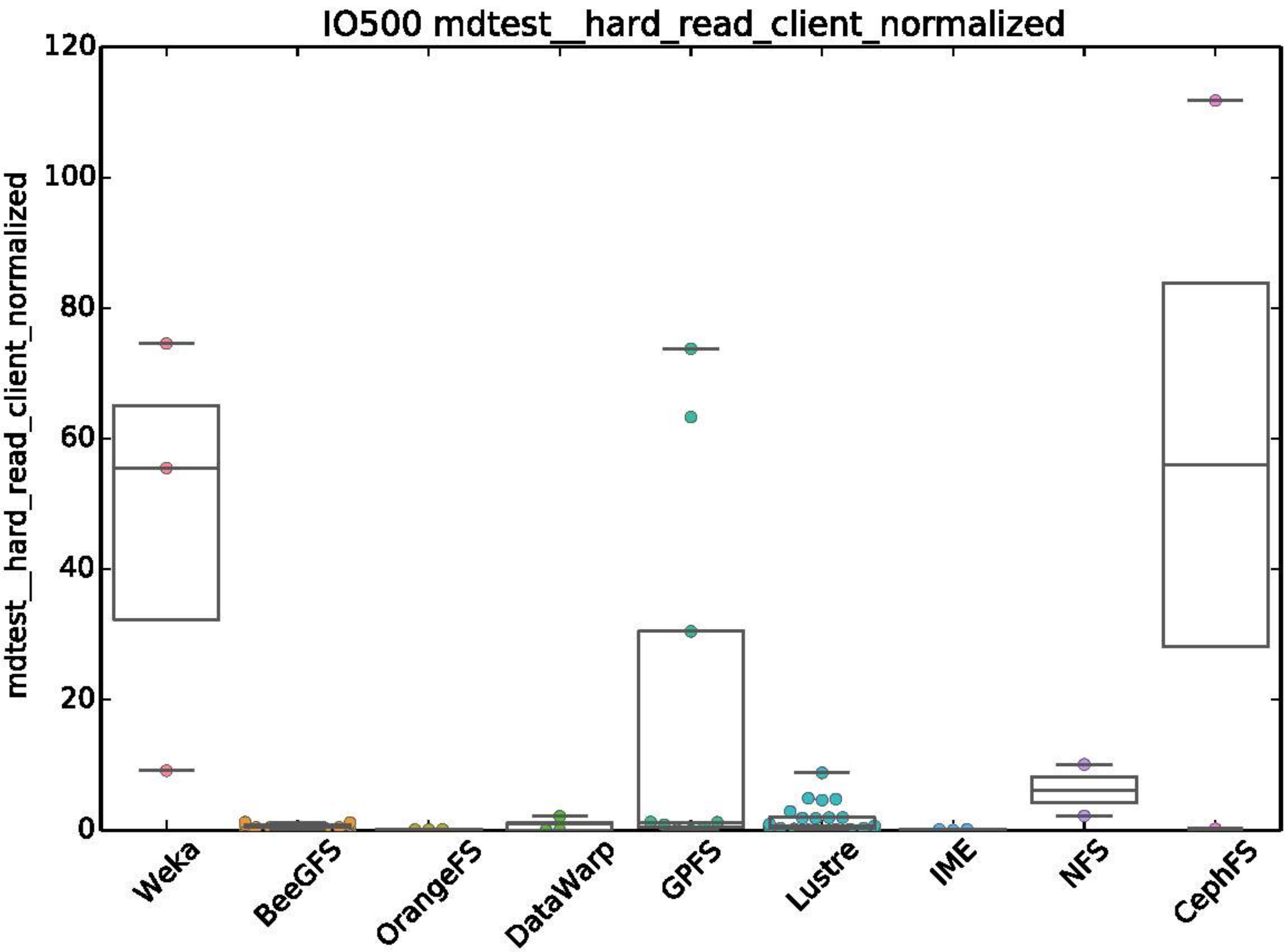

IO500 mdtest__hard_delete_capacity_normalized
mdtest__hard_delete_capacity_normalized
3500
3000
2500
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS



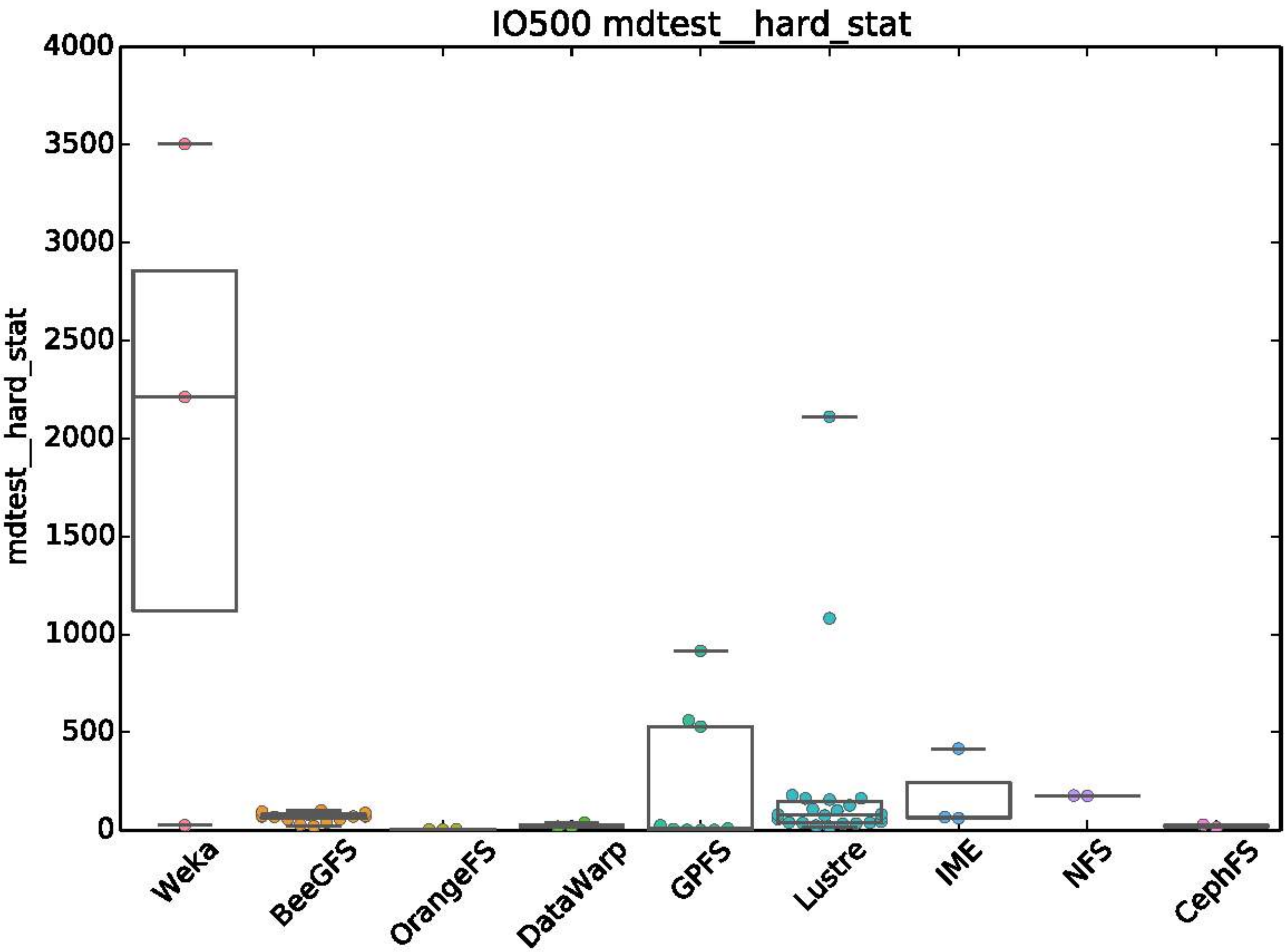
IO500@SC18; Bent

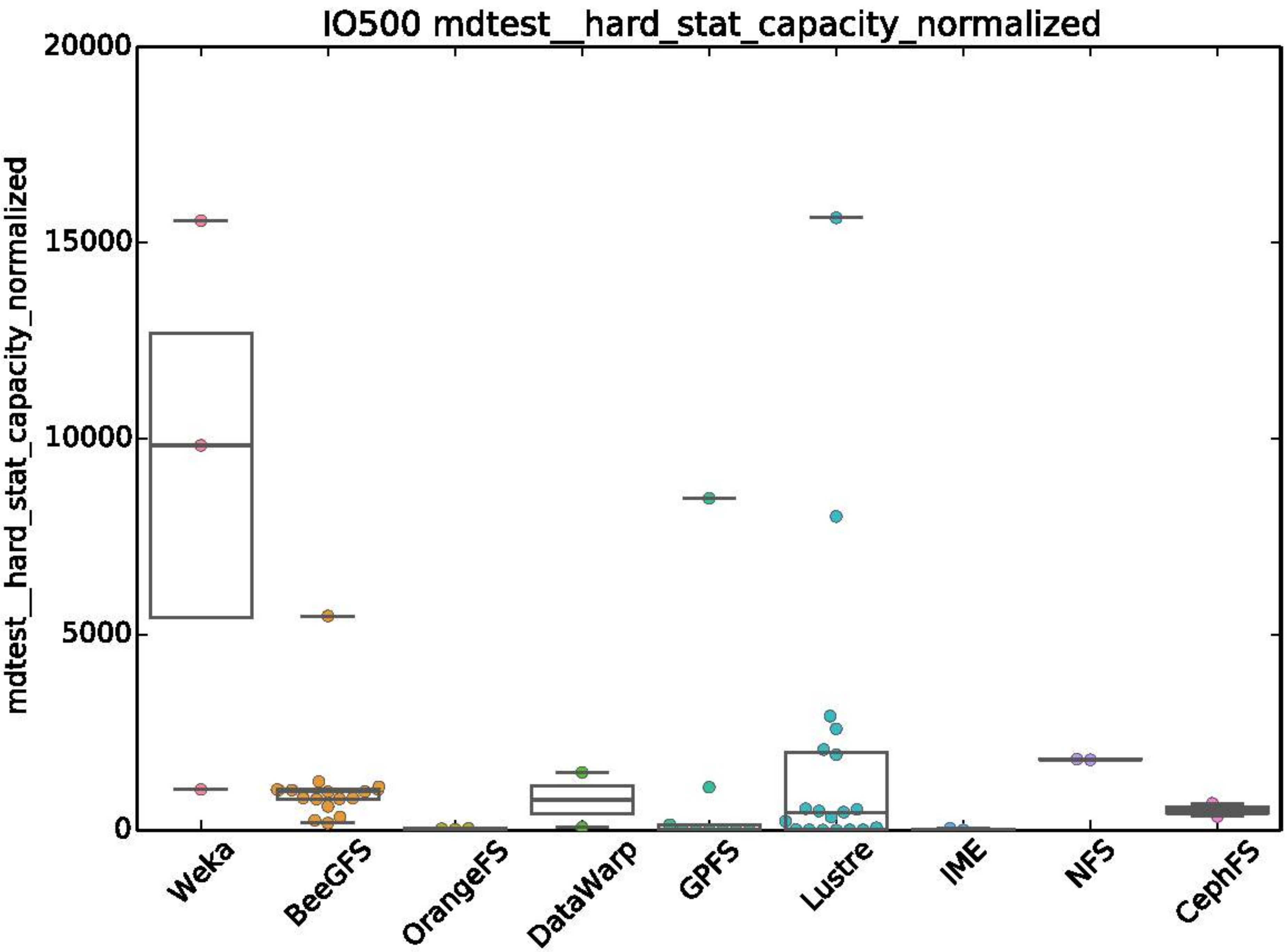

IO500 mdtest__hard_read
mdtest__hard_read
20000
15000
10000
5000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS


IO500 mdtest__hard_read_capacity_normalized
mdtest__hard_read_capacity_normalized
200000
150000
100000
50000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

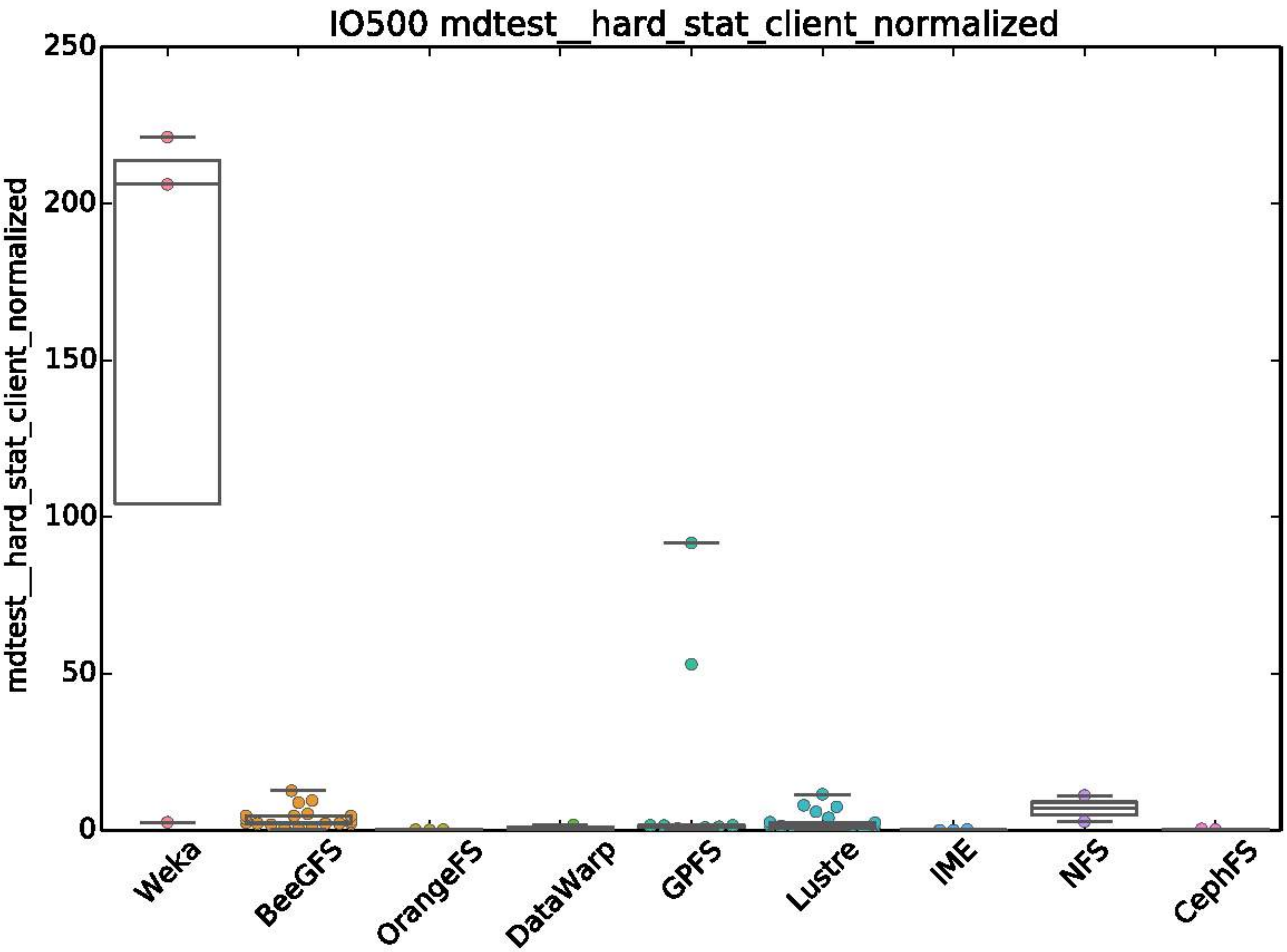

IO500 mdtest__hard_read_client_normalized
mdtest__hard_read_client_normalized
0
20
40
60
80
100
120
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

IO500

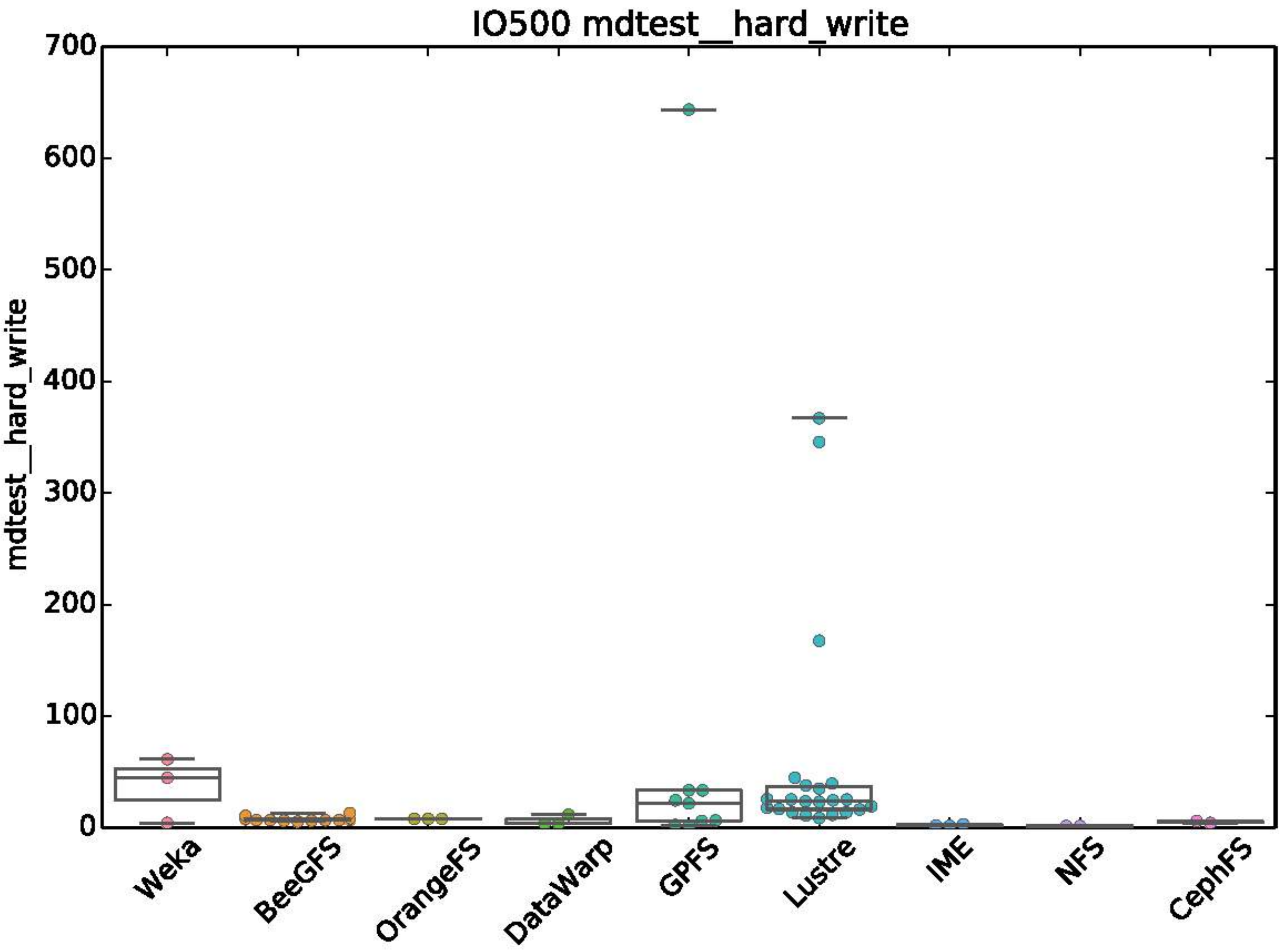

IO500 mdtest__hard_stat
mdtest__hard_stat
4000
3500
3000
2500
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

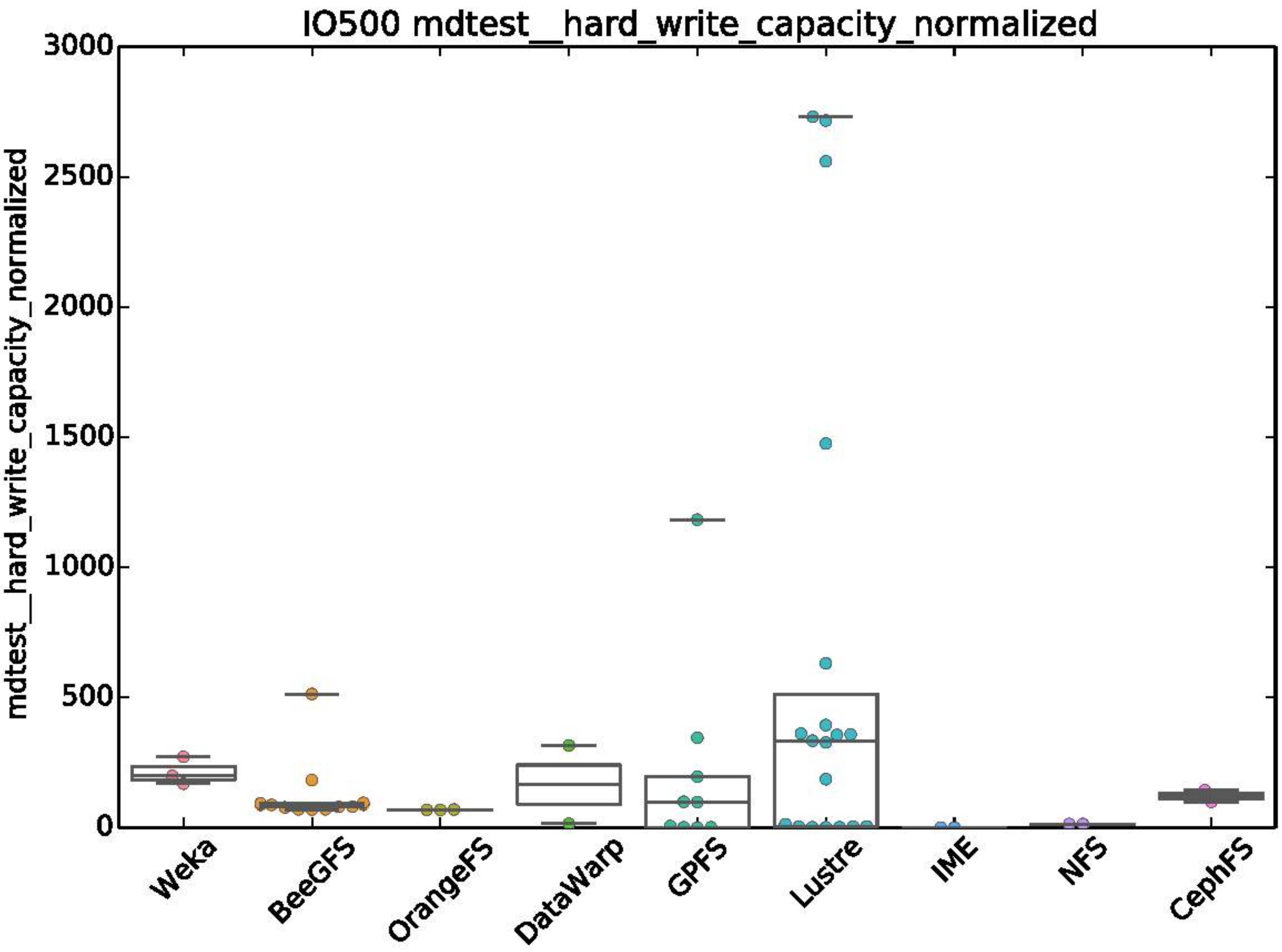

IO500 mdtest__hard_stat_capacity_normalized
mdtest__hard_stat_capacity_normalized
20000
15000
10000
5000
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

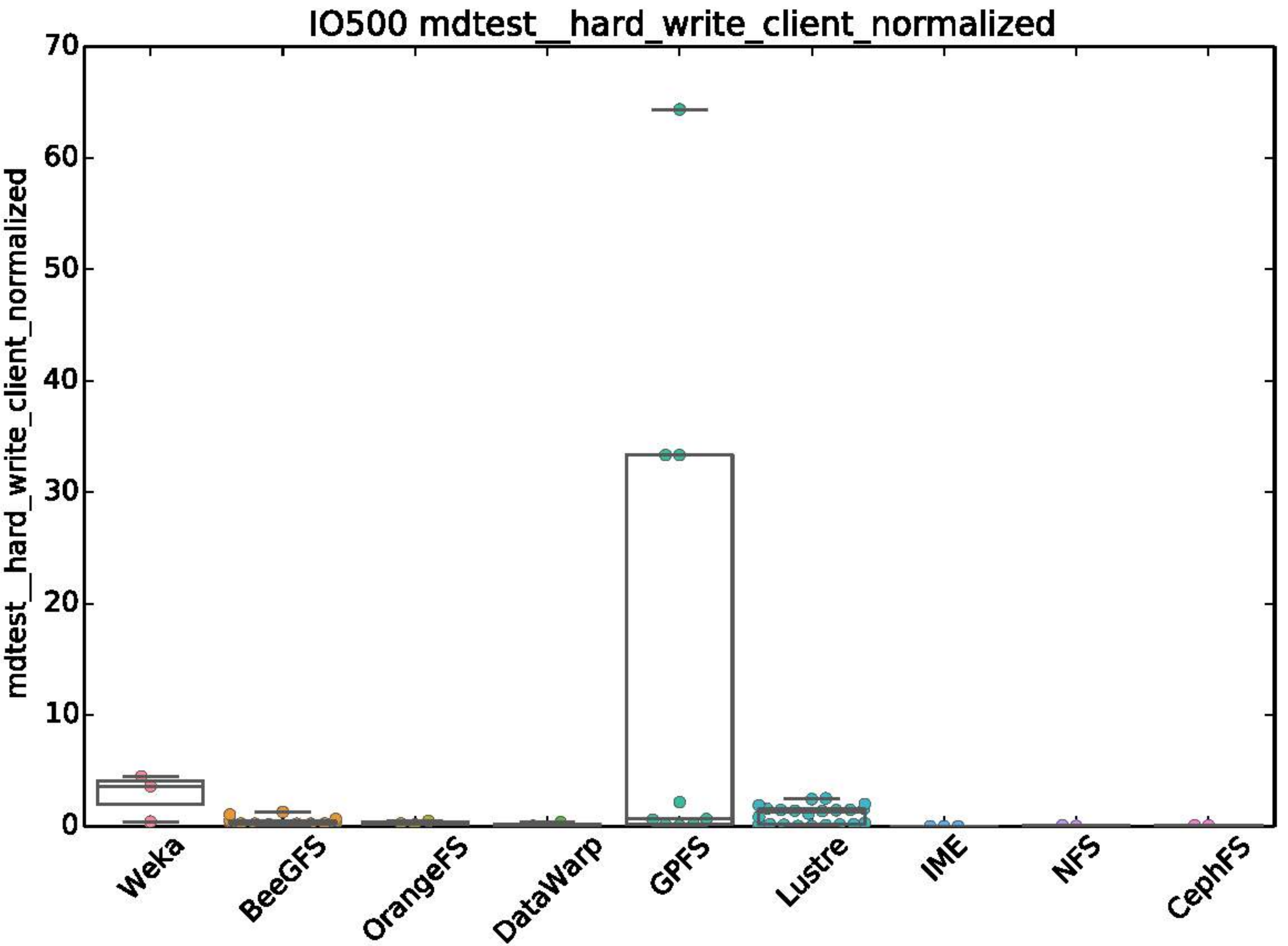
IO 500


IO500 mdtest__hard_stat_client_normalized
mdtest__hard_stat_client_normalized
250
200
150
100
50
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

IO500


IO500 mdtest__hard_write
mdtest__hard_write
0
100
200
300
400
500
600
700
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS


IO500 mdtest__hard_write_capacity_normalized
mdtest__hard_write_capacity_normalized
3000
2500
2000
1500
1000
500
0
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS


IO500 mdtest__hard_write_client_normalized
mdtest__hard_write_client_normalized
0
10
20
30
40
50
60
70
Weka
BeeGFS
OrangeFS
DataWarp
GPFS
Lustre
IME
NFS
CephFS

See you at ISC’19 for the fourth IO500 List!