Tracking user-perceived I/O slowdown via probing

Tracking User-Perceived I/O Slowdown via Probing

Julian Kunkel[1] and Eugen Betke[2]

[1] University of Reading–j.m.kunkel@reading.ac.uk
[2] DKRZ – betke@dkrz.de

Abstract. The perceived I/O performance of a shared file system heavily depends on the usage pattern expressed by all concurrent jobs. From the perspective of a single user or job, the achieved I/O throughput can vary significantly due to activities conducted by other users or system services like RAID rebuilds. As these activities are hidden, users wonder about the cause of observed slowdown and may contact the service desk to report an unusual slow system.
In this paper, we present a methodology to investigate and quantify the user-perceived slowdown which sheds light on the perceivable file system performance. This is achieved by deploying a monitoring system on a client node that constantly probes the performance of various data and metadata operations and then compute a slowdown factor. This information could be acquired and visualized in a timely fashion, informing the users about the expected slowdown.
To evaluate the method, we deploy the monitoring on three data centers and explore the gathered data for up to a period of 60 days. A verification of the method is conducted by investigating the metrics while running the IO-500 benchmark. We conclude that this approach is able to reveal short-term and long-term interference.

Keywords: Parallel File Systems; Performance Analysis; Latency

1 Introduction

HPC centres are highly motivated to improve efficiency and utilization of their systems. The estimation of the current system state is often based on a few metrics, which are continuously observed by a monitoring system.

Although, basic metrics are sufficient for identification of the most typical problems, they are less suitable for special tasks, e.g., one major difficulty for the file system state assessment is that many I/O metrics depend on both: application behavior and file system state. In case of non-obvious I/O issues, i.e., when all metrics looks good except I/O performance, it's not easy to say what is the cause just by looking at them. Understanding the quality of the file system health and quantifying its performance is another challenge. Users may wonder if the subjective experienced performance ("today the system is very slow") is actually true and, particularly, when jobs are aborted since the I/O phases take

The final publication is available at Springer through the DOI 10.1007/978-3-030-34356-9_15

subjectively longer than normal[3]. This may lead to additional queries for the help-desk – which might be busy to resolve an issue. Knowing that the observed file system behavior is monitored and already classified as "slow" due to a degraded system state has a prospect to help users (but also staff) to understand the situation and reduce the queries.

To be able to identify the misbehaving component, it is beneficial to have a reliable metric that refers to file system only, i.e., it must be independent from any application. It would be also beneficial to quantify current file system state and to measure reaction to different I/O workloads. The latter specially useful to assess recurring I/O-intensive tasks such as backup of data.

The **key contribution** of this paper is the introduction of a systematic I/O performance monitoring using a probe with low intrusion and the investigation of means to derive a user-understandable slowdown factor.

The paper is structured as follows: First, we discuss related work in Section 2. Next, in Section 3, we discuss the methodology that we use for this work. The experimental results are presented in Section 4. Finally, the work is concluded in Section 5.

2 Related Work

There are various monitoring systems for cluster activities. Darshan [2, 3] is an open source I/O characterization tool for post-mortem analysis of HPC applications' I/O behavior. Its primary objective is to capture concise but useful information with minimal overhead. Statistics are captured in a bounded amount of memory per process as the application executes. When the application shuts down, it is reduced, compressed, and stored in a unified log file. Utilities included with Darshan can then be used to analyze, visualize, and summarize the Darshan log information. Because of Darshan's low overhead, it is suitable for system-wide deployment on large-scale systems. In this deployment model, Darshan can be used not just to investigate the I/O behavior of individual applications but also to capture a broad view of system workloads for use by facility operators and I/O researchers.

In [9] Uselton and Wright claim that conventional I/O performance analysis based on transferred data volume and bandwidth doesn't provide reliable information about the state of file systems. They investigate a new metric called file system utilization (FSU) that takes into account the number and the size of disk I/O operations. A mathematical I/O transaction model brings all parameters together. On NERSC's Hopper Cray XE6 system they detect a busy file system with FSU, even if I/O performance indicates an idle state. This approach requires a complicated setup, that captured and collects I/O data from client and server nodes.

A machine learning approach for anomaly detection was investigated by Tuncer et al. in [8]. This approach targets two aspects. First, data reduction

[3] When the specified job walltime limit is hit, jobs are terminated.

by mapping large raw time series to a few relevant statistics. Second, automatic anomaly detection and classification with machine learning algorithms. With sufficient data, a model can be adapted to several anomaly types, e.g. hardware issues, memory leaks, system health status. In the experiments on a HPC cluster and a public cloud the resulting ML-model a high accuracy (F-score higher than 0.97). Probably, the biggest challenge of this approach the machine learning technology. Creation of well trained models requires a lot of expertise in this area.

The LASSi Tool [7] captures I/O behavior and computes risk metrics that are related to application's I/O behavior. While this is similar to the goal of this approach, the interpretation of the synthetic LASSi metrics is difficult.

Sometimes developers or users utilize the principle of performance regression testing, i.e., to run an application periodically and measure the performance development while developing the code further. Typically this serves the purpose of identifying introduced performance issues but it is sometimes also used to identify slowdown in the system. For instance, in [10] the authors run benchmarks with Jenkins periodically to monitor system health. In [6], IOR is run daily to track the performance behavior. Additionally, performance of I/O motifs is analyzed over time.

This information is targeted to the data center staff and able to track changes in long-term system behavior (e.g., after updates), while our probing approach addresses the user side focusing on giving feedback of the perceived system load in a timely fashion.

3 Methodology

To quantify the user-slowdown, we periodically gather the response time from the client side and then apply statistics to reduce data points to meaningful metrics.

3.1 Probing

An accurate analysis of client-perceived response time can be done by running probes on a compute node periodically; a probe executes a small I/O operation. We assume that the response time of the probe is representative for similar I/O operations during the time of execution.

As the file system performance depends on the type of the operation performed, in this paper, we cover several data and metadata patterns as follows: **Data I/O**: a block of 1 MB is read and written each at different random locations in a large file. **Metadata**: the operations create, stat, read, and delete of small files (3901 Byte) are executed on a pool of files, i.e., the number of files of the pool remains identical as each iteration accesses the oldest created file and deletes it, and then creates a new file.

These six operations are executed sequentially periodically by the probing tool and the response time is measured for each operation individually. We use

the `dd` tool to measure data I/O latency and for metadata the MDWorkbench [4] – this test allows to iterate on a working set for regression testing. The purpose of this process is not to mimic the behavior of a parallel application but to probe the system performance from the user-side – this is the performance a user would see.

Discussion. The frequency of the polling could be adjusted to minimize the impact on the file system – we use 1s as the generated load with 2 MB/s and 5 metadata operations/s is negligible. Instead of running sequentially, the tests could be run in parallel or by spawning one thread for each operation. However, this would increase the load of the file system when the file system is already overloaded and makes it more difficult to assess the results as the measurements interfere with each other.

Running such a tool requires additional compute resources and storage capacity. Firstly, to reduce the impact of caching, it is required that a sufficiently large file is pre-created (exceeding the memory of the client) and that the pool of files is sufficiently large. Additionally, a file should be spread across all servers and storage media. Depending on the system configuration, a random I/O operation may access only a single server or storage media. As the probing frequency is very high, we claim this is still representative for storage systems with 100 servers – to increase the accuracy for large systems, multiple I/O probes could be executed, one per server.

The probing tool could run as a service on a maintenance node, on compute nodes, or on shared/interactive nodes. As additional I/O operation performed on the probing server (e.g., by users) influences the response time, for most accurate results, the probing tool should be run on a compute node exclusively – which implies some costs. Our experiments have shown that deployment on a shared node still produces meaningful results.

3.2 Data Reduction using Statistics

When measuring small I/O requests, the response time is expected to vary frequently between two consecutive measurements but also statistics of longer sequences of measurements differ. The former is caused by short events like concurrent accesses, cache misses, or network congestion, while sequences of measurements are impaired by the long-term effects we are interested in.

In order to report meaningful slowdown metrics for longer intervals, high-frequent changes must be dampened. A typical approach is to compute the mean value in an interval. The arithmetic mean, however, is sensitive to outliers, therefore, we explore the use of various statistics particularly, the median, and the 90% and 95%-quantiles. Reporting the statistics for a quantile allows to understand the fraction of I/O operations that are slowed down in a certain interval, e.g., reporting a waiting time of 0.1s for the 90%-quantile means that 90% of the operations are faster and 10% are slower than 0.1s. Depending on the intended service level agreement with users, this is more appropriate than reporting high means caused by single stragglers.

3.3 Computing the Slowdown

Generally, we could define the user-perceived slowdown of a particular operation naturally as the observed response time of the operation divided by the expected response time: $s = \frac{t_{\text{observed}}}{t_{\text{expected}}}$. Thus, the reported slowdown is a factor by which I/O is expected to take longer. An issue is the robust determination of t_{expected}. A production system is typically experiencing slowdown over an empty system. This might confuse users to see that a system is typically slow by 3x, sometimes by 5x. Therefore, we define t_{expected} as median of all observed values (for the specific operation)[4]. The reported slowdown is then computed based on the statistics for each interval.

4 Evaluation

4.1 Test Systems

JASMIN is the national data analysis facility of the UK for the European climate and earth system modelling community [5]. For each file system a 200 GB file is precreated a pool of 200k files for MDWorkbench. The script is run exclusively on one node and after 20 hours the job is restarted; as there are many nodes available, typically a host runs only one or two 20 hour periods during the 60 days of recording.

Archer is the UK national supercomputing service[5] consisting of a 4,920 node Cray XC30 utilizing the Sonexion Lustre storage. The file system used for testing provides 1.6 PB of capacity on 14 OSSs and 56 OSTs. For each file system a 200 GB file is precreated and a pool of 200k files for MDWorkbench. The jobs are executed on one of two interactive nodes that are shared with the users.

Mistral the High Performance Computing system for Earth system research (HLRE-3), is DKRZ's supercomputer. The HPC system has a peak performance of 3.14 PetaFLOPS and consists of approx. 3,300 compute nodes, 100,000 compute cores, and 266 Terabytes of memory. The total capacity of two Lustre file systems attached to the system is 54 PB (lustre01: 21PB and lustre02: 33 PB). All components are using FDR Infiniband interconnect.

For each file system a 1.3 TB file is precreated and a pool of 100k files for MDWorkbench. During the experiments the I/O-Probing tool runs on a login node, sharing the system with other users. The tool is executed on an interactive node that is shared with the users.

[4] The value could be updated periodically in a sliding window to cover the typical operational conditions or it could utilize other statistics than the median.

[5] `http://www.archer.ac.uk`

4.2 Probing Tool

Our first measurements are done with a BASH script, that implements the behavior from Section 3.1. On systems with older kernels, we encountered an issue with frequent forking of probes in new processes. Over long time period, it causes high CPU load and makes compute nodes unresponsible. Therefore, we created a tool – I/O-Probing [6] that exhibits the same behavior; it is a single process solution written in C programming language. With corresponding program parameters it emulates the behavior of the script. It is using the sophisticated functions from MDWorkbench to get reliable response times and to calculate statistics. In the read/write test, the I/O-Probing tool accesses a large file with random patterns using POSIX `read64()/write64()`.

4.3 Timeseries of Individual Measurements

First, we investigate the gathered data for JASMIN on various levels of detail starting from individual measurements. Figure 1 shows the response time of each individual probe on two file systems for a duration of 10 minutes. It can be observed that the performance on the home directory is very predictable for all operations, as this is a PureStorage system and not used for parallel data production, this is expected. The 1 MB read operations express two bands of accesses, one around 0.1ms and one around 10ms – the latter are caused by disk accesses while the former is caused by caching on client or server[7].

The work file system is at the beginning also robust but after 1.5 minutes, the behavior changes, a fraction of operation is now 10-100 times slower (stat and delete remains low). As only isolated operations are slow, this is likely caused short-period load on the data and metadata servers.

4.4 Host variability

As hosts are typically connected identically to the storage, it is expected that the reported response time is independent from the node, where the script is executed. This is important as we assume that the observed response time on the probing host is similar to the response time on other (non-monitored) hosts.

We verified this by running the script on JASMIN in parallel on five different nodes for a period of one hour covering at least 2,000 measurements of each metric per host. The obtained statistics are visualized in Figure 2 as boxplot. Most metrics are indistinguishable for the different hosts and the systematic difference in behavior between home and work directory is observable. However, the first and third quartile for the write and read performance are slightly different and so is the median for Host 5. Still, we assume that the metrics are robust as different nodes observe them similarly.

[6] `https://github.com/joobog/io-probing`
[7] To minimize this, the precreated file size could have been increased.

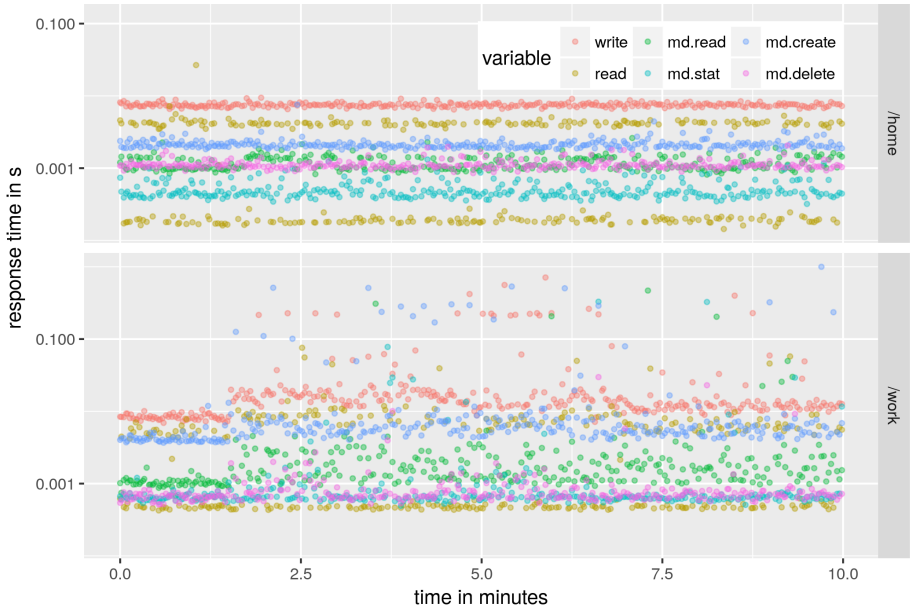


Fig. 1: Jasmin every data point for 10 minutes from one node

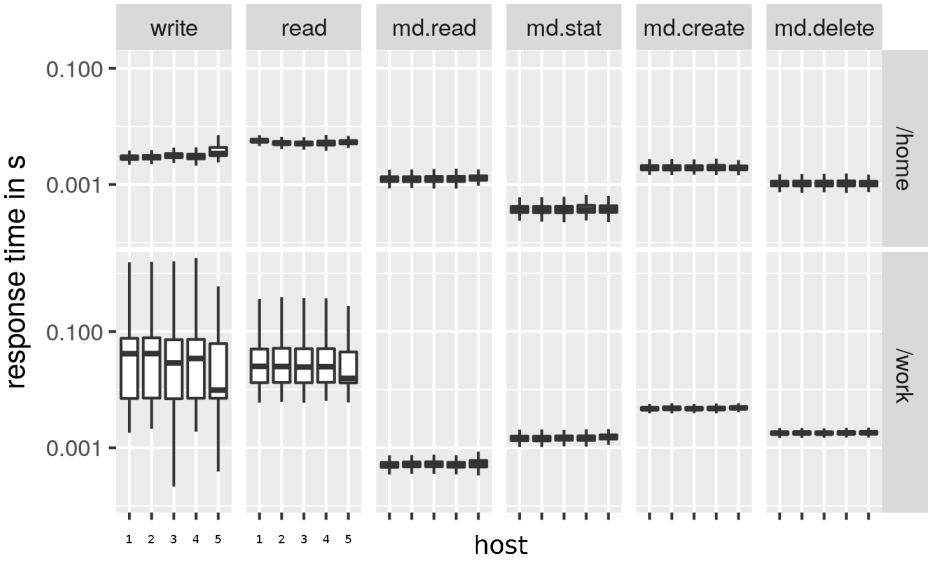


Fig. 2: Jasmin boxplot statistics from five different hosts

4.5 Understanding Application Behavior – The IO-500

The IO-500 [1] is a benchmark suite that establishes I/O performance expectations for naive and optimized access; a single score is derived from the individual measurements and released publicly in a list to foster the competition.

The IO-500 is built on the standard benchmarks MDTest and IOR[8] and represents various metadata and data patterns:

- IOREasy: Applications with well optimized I/O patterns
- IORHard: Applications that require a random workload
- MDEasy: Metadata and small object access in balanced directories
- MDHard: Small file data access (3901 bytes) of a shared directory
- Find: Locating relevant objects based on name, size and timestamp

The workloads are executed in a script that first performs all write phases and then the read-phases to minimize cache reuse. To investigate the behavior of the risk for running applications, we executed the IO-500 benchmark on 100 nodes on Archer resulting in a total IO-500 score of 8.45.

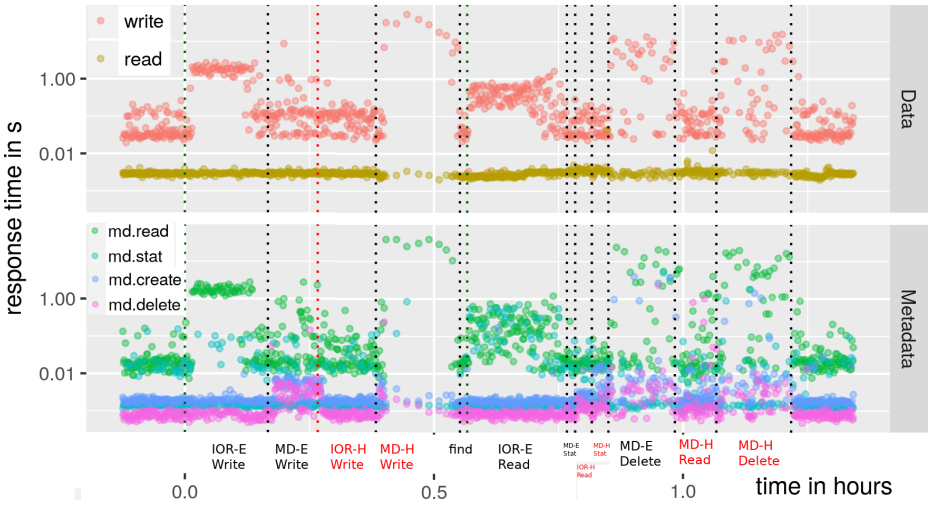
As they are designed to be part of the user-side monitoring, the probes are run concurrently with the benchmark (and any other I/O operation) and, in case of server-side contention, the latency is expected to increase. The response time of the probe is shown in Figure 3(a) correlating directly to the slowdown in Figure 3(b) – note that the slowdown factor is still computed based on the median for the 30 day period. In many cases, the influence of a metadata or data heavy benchmark is directly observable; for metadata benchmarks, the response varies more. The biggest impact has the MDHard benchmark leading to a 1000x slowdown followed by the 100x slowdown of IOREasy.

To show the impact of data reduction by applying statistics, Figure 3(c) computes the slowdown by using the mean for 60s intervals reducing the noise and short-term influences. As most phases of the IO-500 are several minutes, these remain well represented. A key difference is that the mean of the `stat()` call of a 60s period still behaves as the mean for all stat calls while individual calls are delayed.

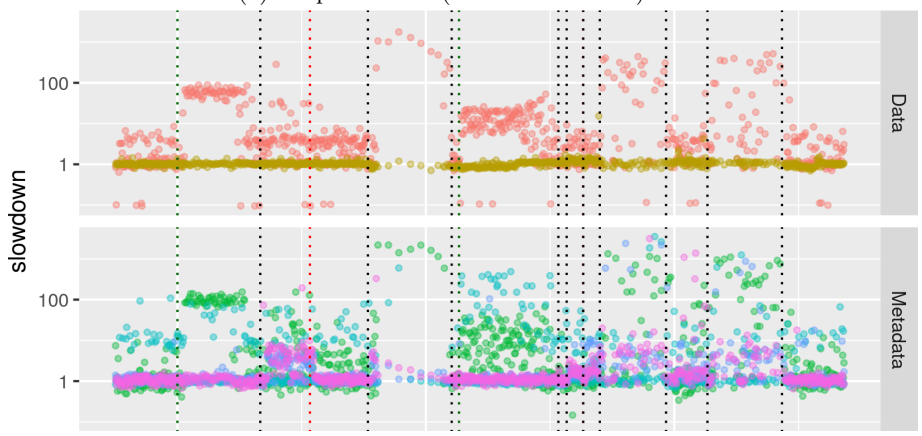
4.6 Long-Period

Next, we investigate the measured statistics for a 60 day period on JASMIN, 30 day period on Archer and 18 day period on Mistral. Metrics for data and metadata are aggregated for intervals of 4 hours; this reduces the data points for visualization, and allows to investigate long-term influences on performance. The observed performance statistics for various access types and the three data centers are shown in Figure 4 and selected timelines in Figure 5. Note that the observations differ significantly between the three data centers and between file systems.

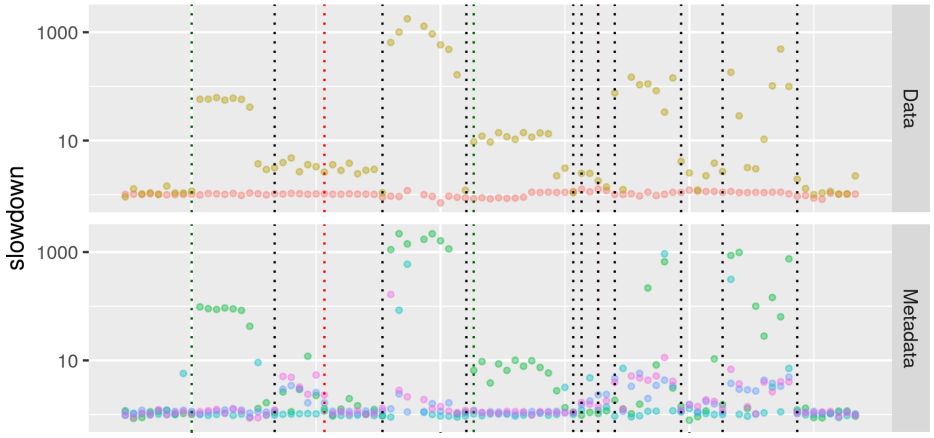
[8] `https://github.com/hpc/ior`



(a) Response time (all measurements)



(b) Slowdown (all measurements)



(c) Slowdown (60s mean statistics)

Fig. 3: IO-500 on Archer with annotated phases

JASMIN. Write looks actually very similar to read and is therefore omitted. The boxplot (Figure 4a) shows the distribution of the statistics for the 60 days; indicating that all statistics including mean vary by three orders of magnitude for work and by 10-100x for home. The interquartile distance indicates that 50% of time the statistics are rather robust.

The timeline in Figure 5a shows that consecutive statistics typically exhibit similar behavior indicating that some longer-running I/O-heavy activity creates interference. The 95% and 99% quantiles are less robust but tell us something about the distribution of performance; the slowest operation of a 4 hour period takes at least a few seconds but also sometimes 10-100s!

The metadata statistics (Figure 4b) indicates that the metadata performance varies stronger. The timeline reveals that consecutive accesses often achieve similar performance (not shown due to space constraints).

Archer. The timeline for metadata probing is shown in Figure 5b (the data timeline is not so interesting for work). A weekly slowdown of the small read and delete operations is observable (similarly for the large read accesses too, not shown) on home, this is likely due to the backup. On work, the statistics of a 4 hour interval stays robust, except for the first days and the last few days.

The statistics for some 4h intervals are 100x slower than typically (see Figure 4c, Figure 4d). The read statistics with a median of 5 ms indicates that data is often served not from the HDD; the 200 GB files still lead to cache hits.

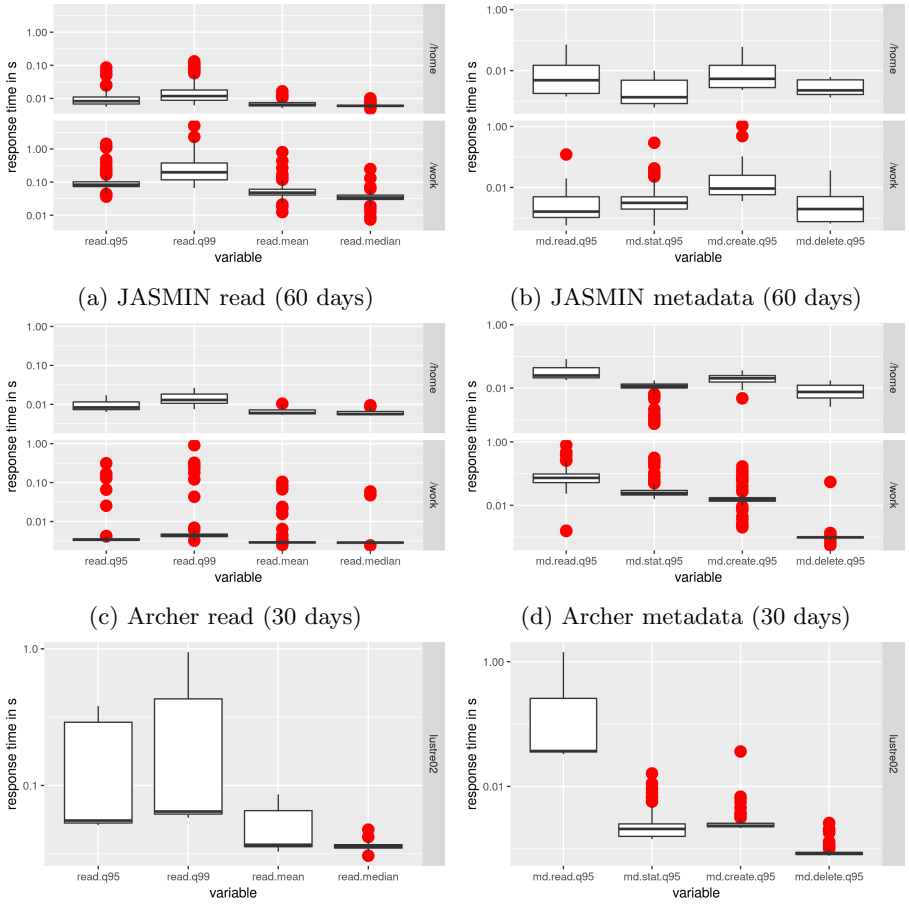
Mistral. During the experiment we started two instances of the I/O-Probing tool, for each file system one. On the lustre01 file system, we couldn't observe anomalies (results are omitted). Generally, the data and metadata performance is worse compared to the two other data centers (compare Figure 4).

On lustre02, there is one 27x slowdown compared to typical performance for read (see Figure 5c). It takes almost 6.5 days for the system to recover from the I/O load. One reason could be an I/O intensive chained job, that was running on a shared SLURM partition as shared partitions allow for a runtime up to 7 days. It could also have been a degraded file system, e.g., server outage in the active-active fail-over.

4.7 Slowdown

The computation of a slowdown factor and presentation to the users was the main goal of this article. As we discussed in Section 3.3, longer interval length lead to a smearing of short-term effects. In Figure 6, the slowdown for JASMIN and Archer is presented that is computed on a 4 hour and 10 minute interval, respectively (using the median). Note that the normalization by the median leads to cases where the reported performance is actually faster than the median, and thus a slowdown below 1 – which is the normal behavior – is observable. These fast responses could be caused by caching effects.

By looking at the graphs, the expected slowdown of the I/O could be determined, for example, a slowdown of 10 can be seen for read operations on



(a) JASMIN read (60 days)

(b) JASMIN metadata (60 days)

(c) Archer read (30 days)

(d) Archer metadata (30 days)

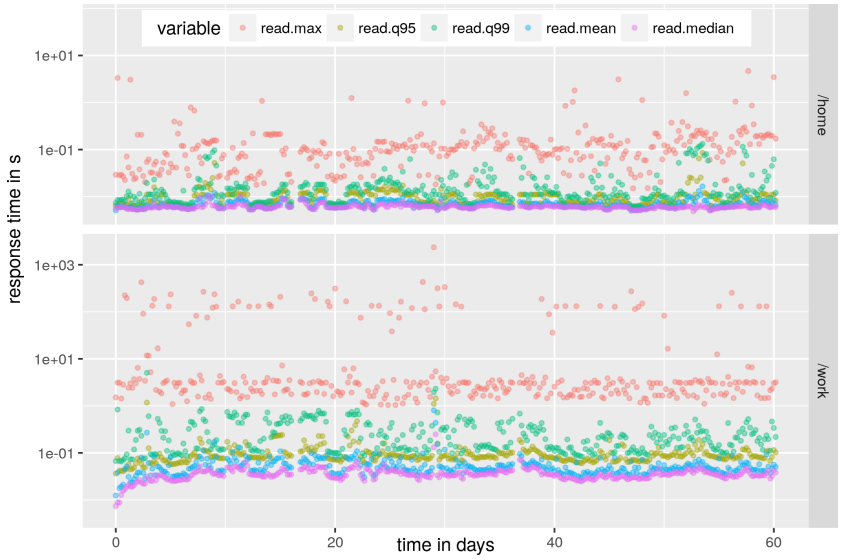
(e) Mistral read (18 days)

(f) Mistral metadata (18 days)

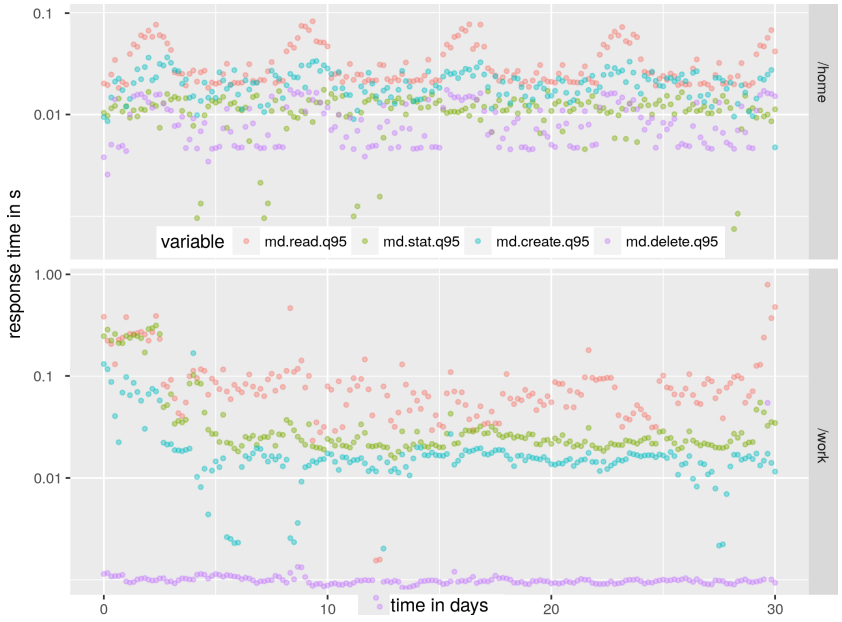
Fig. 4: Statistics for 4 hour intervals

JASMIN for several 4 hour periods. Remember, the slowdown for a given type of operation quantifies the expected slowdown compared to normal operation. Thus, for I/O, the reading/writing is expected to take x times longer than normal. This is expected to be true for small application runs. However, I/O intense hero runs are expected to cause the I/O slowdown. The impact of running them on other applications is the cause of the experienced slowdown factor.

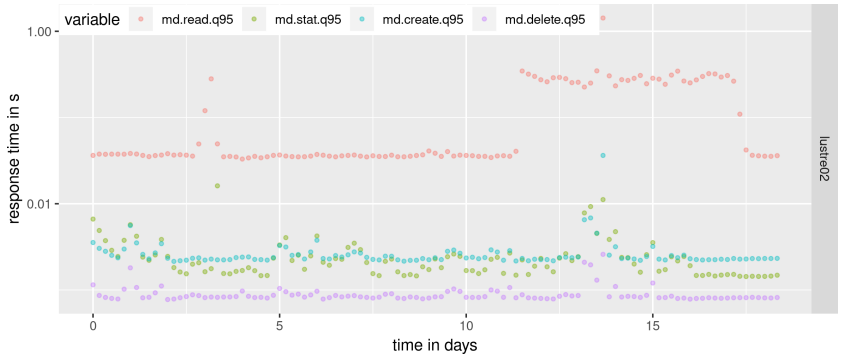
Generally, the effects on JASMIN take longer and are well visible, while the work directory on Archer exhibits mostly short-term bursty behaviors that is invisible in 4 hour intervals. The home directory exhibits a slowdown of 3 during the weekly service activities. We conclude that by setting the interval length appropriately, short-term and long-term influences can be detected and communicated to the user.



(a) JASMIN read timeline



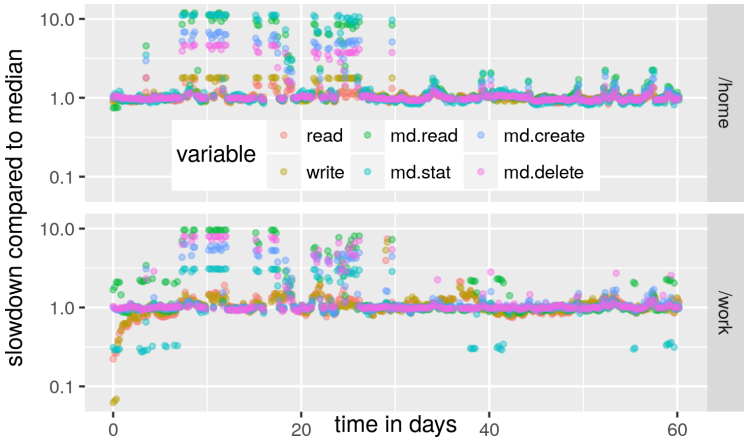
(b) Archer metadata timeline



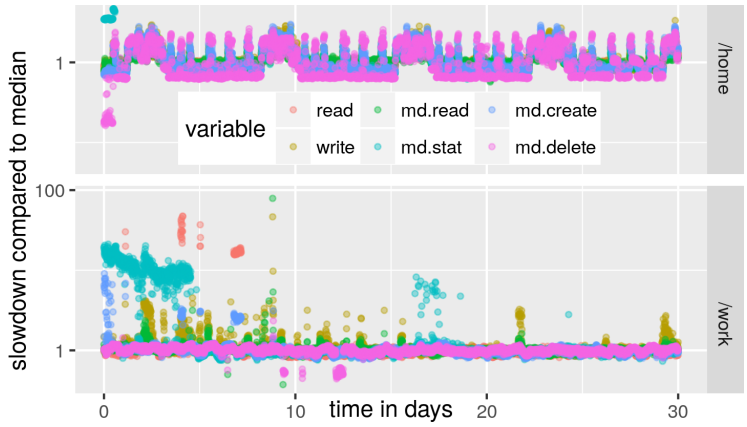
(c) Mistral metadata timeline

12/14

Fig. 5: Timelines for Data or Metadata – statistics for 4 hour intervals



(a) JASMIN, computed on 4 hour intervals



(b) Archer, computed on 10 minute intervals

Fig. 6: Computed slowdown compared to median

5 Conclusion

In this article, we introduced an approach to monitor the system performance and derive the user-experienced slowdown by constantly running small I/O probes. This allows users and data center experts to investigate and track system performance over time but also to quantify the slowdown compared to normal operation. The slowdown factor is expected to correspond to the delay of parallel applications for data bound and metadata bound applications. The data could be feed into a monitoring system such as Grafana allowing stakeholders to investigate the slowdown in a timely fashion – for instance, helpdesk employees can answer the question if the storage is currently overloaded.

A key issue is the statistical reduction of the sampled data, depending on the selection of the selected metric, the effect on the user can be investigated.

We demonstrated the effectiveness of this approach on the IO-500 workload and showed the high variability of 4-hour statistics for three data centers. We were particularly surprised to see small individual I/Os that can take up to 100s.

In the future, we will combine this approach with additional server-side data and user-utilization and perform long-term studies.

Acknowledgements

This work was supported by the UK National Supercomputing Service, ARCHER funded by EPSRC and NERC. We thank the German Climate Computing Center (DKRZ) for providing access to their machines to run the experiments.

References

1. John Bent, Julian Kunkel, Jay Lofstead, and George Markomanolis. IO500 Full Ranked List, Supercomputing 2018 (Corrected), November 2018. `https://www.vi4io.org/io500/list/19-01/start`.
2. Philip Carns. Darshan. In *High performance parallel I/O*, Computational Science Series, pages 309–315. Chapman & Hall/CRC, 2015.
3. Philip Carns, Kevin Harms, William Allcock, Charles Bacon, Samuel Lang, Robert Latham, and Robert Ross. Understanding and improving computational science storage access through continuous characterization. *ACM Transactions on Storage (TOS)*, 7(3):8, 2011.
4. Julian Kunkel and George S. Markomanolis. Understanding Metadata Latency with MDWorkbench. In *High Performance Computing: ISC High Performance 2018 International Workshops, Frankfurt/Main, Germany, June 28, 2018, Revised Selected Papers*, number 11203 in Lecture Notes in Computer Science, pages 75–88. Springer, 01 2019.
5. B Lawrence, V Bennett, J Churchill, M Juckes, Philip Kershaw, P Oliver, M Pritchard, and A Stephens. The JASMIN super-data-cluster. *arXiv preprint arXiv:1204.3553*, 2012.
6. Glenn K Lockwood, Shane Snyder, Teng Wang, Suren Byna, Philip Carns, and Nicholas J Wright. A year in the life of a parallel file system. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, page 74. IEEE Press, 2018.
7. Karthee Sivalingam, Harvey Richardson, Adrian Tate, and Martin Lafferty. LASSi: metric based I/O analytics for HPC. *SCS Spring Simulation Multi-Conference (SpringSim'19), Tucson, AZ, USA*, 2019.
8. Ozan Tuncer, Emre Ates, Yijia Zhang, Ata Turk, Jim Brandt, Vitus J. Leung, Manuel Egele, and Ayse K. Coskun. Diagnosing Performance Variations in HPC Applications Using Machine Learning. In *High Performance Computing*, pages 355–373, Cham, 2017. Springer International Publishing.
9. Andrew Uselton and Nicholas Wright. A File System Utilization Metric for I/O Characterization. 2013.
10. Joseph Voss, Joe A. Garcia, W. Cyrus Proctor, and R. Todd Evans. Automated System Health and Performance Benchmarking Platform: High Performance Computing Test Harness with Jenkins. In *Proceedings of the HPC Systems Professionals Workshop*, HPCSYSPROS'17, pages 1:1–1:8, New York, NY, USA, 2017. ACM.