NHR-Storage-concept

Hendrik Nolte, Julian Kunkel, Jan Frenzel, Maximilian Knespel,
Matthias Lieber, Marcel Nellesen, Steffen Christgau

April 2023

1 Introduction

Within the NHR "Zukunftsprojekt" Large-Scale Data Management it is investigated, how NHR centers can improve their service for data-intensive projects. These projects projects are distinguished from compute-intensive projects, that they use data-parallelism rather then task-paralellism. However, this also leads to distinct challenges for data-intensive projects. This is specifically the case for storage requirements and usage patterns. One famous example for this are machine learning workloads which in some cases iteratively read in a large amount of small files. But also other challenges with regard to quota management, or data locality can be become important. Within this workpackage, different points of consideration with a focus on storage have been investgated by a specific centre and has then be discussed within the regular project meetings. In the following, these vastly diverse research items are presented.

2 GWDG

2.1 IO-Weather Map

The term weather is generally known as a state of the atmosphere at a particular place and time with regard to certain observables like temperature, wind, cloudiness, or rain. With a similar idea, we want to coin the term "I/O-Weather" which represents the state of an I/O-System at a particular node and time with respect to certain observable performance characteristics. This idea is fundamentally different from the typically employed serverside monitoring of storage systems because it does not contain any information about the I/O weather on a specific node. This makes it difficult to support users when they are suddenly observing a changing behavior of their applications. Therefore, this approach has the unique advantage of accurately representing the performance at the node, i.e., making the performance a user experience observable. However, as useful as this metric is, it is very expensive to actually obtain the data when compared to simpler server-side monitoring.

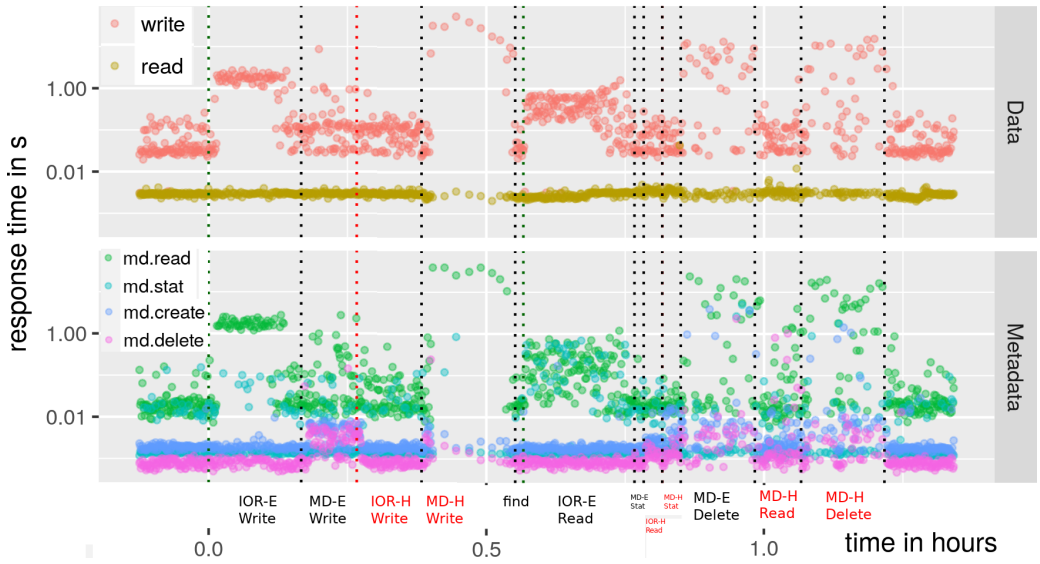

Figure 1: Caption

2.1.1 Measurement Approach

One important distinction of I/O-Weather is with regard to regression tests. In regression tests large-scale I/O Benchmarks, like the IO500, or periodically run on the cluster to check if the originally measured peak performance can be reproduced. These regression tests are therefore extremely useful to detect gradual changes over time or detect a more abrupt performance slowdown due to an update or config change.

I/O-Weather in comparison is the short-term effect. This can be obtained, for instance, by using node-local I/O probing, i.e., small periodic micro-benchmarks to quickly measure the available I/O performance on a single node at a specific point in time. The probing can be done once a second by a cron job executing a bash script. For instance, for the data read/write access with `dd` on a random 1MB block. The metadata operations are measured with *MD-Workbench's* stat, read, delete, and write of a single file per iteration. Other (periodic) alternatives are imaginable.

2.1.2 Example Measurement

One result of such an approach to measure I/O-Weather is shown in Figure 1. Here, the response times in seconds are shown with regard to read and write access for data, and for read, stat, create, and delete for metadata access. One can clearly see the influence of the different phases of the IO500 benchmark on the latencies measured by the small probes. For instance, one can see that data write access times determined by the probe are 100 times slower when it

is executed at the same time when an `ior-easy-write` is running. A similar increase at the same time, can be determined for the metadata read operation. However, a `ior-hard-write` has a lower impact of the I/O-Weather when compared to the `ior-easy-write`. The operation with the highest impact is the `mdtest-hard-write` where an increase of 1000 times can be observed. Generally, the data read operation is much less influenced by the I/O-Weather than the data write. Similarly, for the metadata operations, the create, delete, and stat operations are less affected than the read, while the stat is more affected than the create and delete.

2.1.3 Conclusion

The I/O-Weather can find abnormal I/O behavior on compute nodes which might not be easily extractable solely from the storage servers and can be used to better understand the performance a user experiences.

3 NHR@TUD

3.1 Experiences with the Lustre Storage of latest HPC Installation

NHR@TUD installed a new CPU-based HPC cluster (Barnard) and brought into user operation in October 2023. Part of the procurement was a new DDN Exascaler 6 storage system which is used for working data (scratch) and midterm archive, both running Lustre FS. The scratch file system consists of 7 DDN ES400NVX2 equipped with a total of 770 TiB NVME capacity and 21 PiB spinnig disk capacity. NVMEs are used as hot pools for storage tiering (and to a smaller extend for metadata). Hotpools should especially accelerate random accesses. The performance with IOR easy is about 400 GiB/s write and 600 GiB/s read. Wih mdtest easy write the file systems achieves more than 1.4 MIOPS. However, with hot pools we experienced issues in the automatic migration of data between from NVMEs to HDDs and the resilience of the OSTs against IB network instabilities, which both required updates by the vendor. It will be interesting to see if the hot pool chache enables a more efficient, realiable, and user-friendly I/O environment compared to an explicit staging with a relatively small burst buffer and a large parallel filesystem.

3.2 Improving I/O for many small files

Recently, machine learning applications and experiments became popular. Datasets for training machine learning models are available as compressed archives containing many small files, e. g., ImageNet. However, accessing large amounts of small files generates new challenges in the field of data storage and data access due to many metadata operations. Most storage systems are optimized for accessing comparably few large files. Ratarmount [KRH+24] bridges this gap by letting a user mount a compressed archive as an additional file system. To the

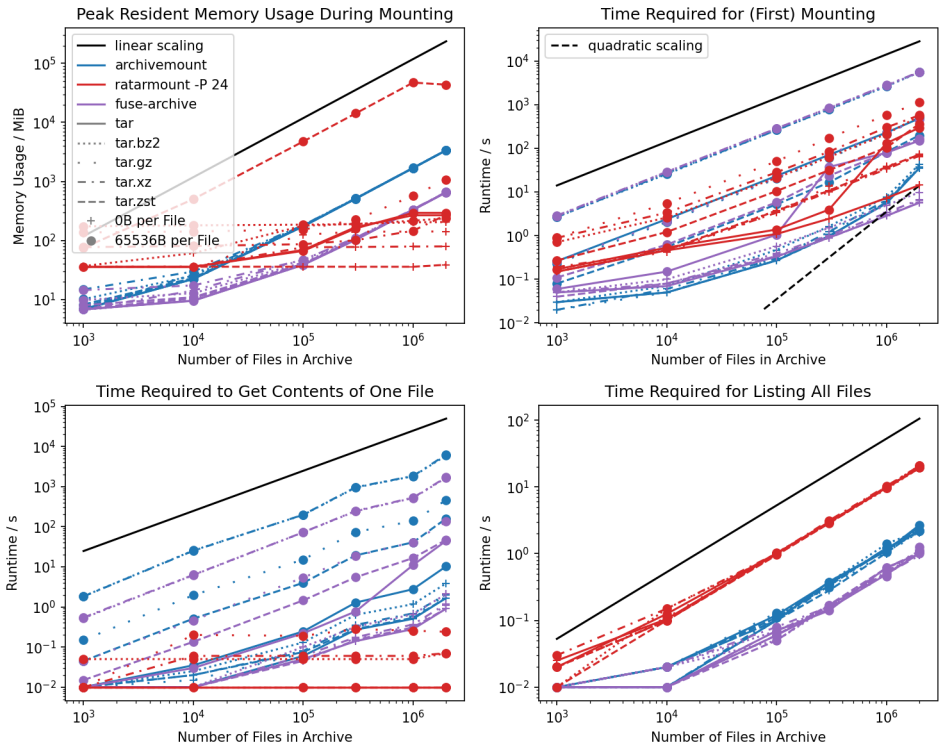

Figure 2: Ratarmount measurements 1

data storage layer of the normal file system, e. g., Lustre, only the archive is accessed, so that expensive and slow metadata operations are avoided. However, the user can still access the files as if the archive would have been extracted first.

3.2.1 Measurements

The capabilities of Ratarmount are depicted in fig. 2 and fig. 3. The diagrams in fig. 2 from top left to bottom right show the memory usage, the time required for (first) mounting, the time required to get the contents of one file and the time required for listing all files depending on the number of files in the archive. In these diagrams, lower values are better. Ratarmount outperforms the other approaches in all except the time required for listing all files for large numbers of files in the archive. Note that ratarmount also shows better scaling behavior than all other approaches. The diagram on the top right represents the first mounting phase. In subsequent mount operations, ratarmount does not add extra time.

The diagrams in fig. 3 from top left to bottom right show the peak resident

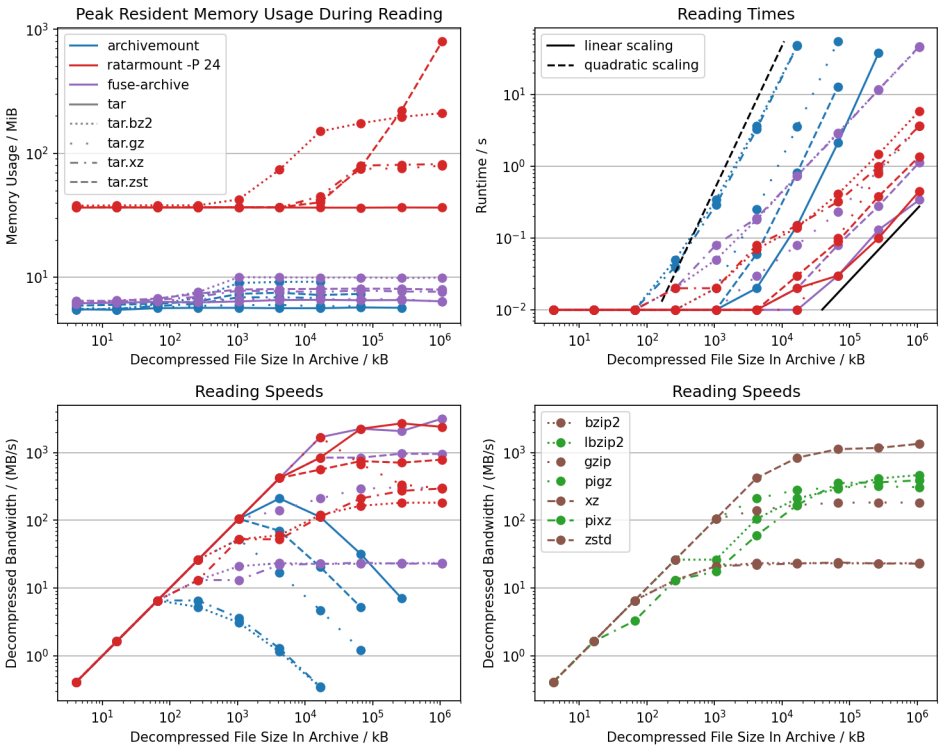

Figure 3: Ratarmount measurements 2

memory usage, the reading time, the decompressed bandwidth depending on the decompressed size of the files in the archive. For the bandwidth measurements, higher values are better, whereas for the other measurements lower values are better. The diagrams depict the high decompressed bandwidth and low reading time of archives with ratarmount compared to the other programs in the test. There are two reasons for the faster performance. Firstly, ratarmount has a novel parallelized backend for gzip decompression [KB23] and a parallelized backend for bzip2 decompression. Secondly, same as `fuse-archive`, it avoids seek -backs for each 4 KiB block requested via FUSE, which is the reason for the decreasing performance for larger archive sizes of `archivemount`.

3.2.2 Conclusion

Ratarmount shows higher speed and better scaling behavior for large archives than other solutions such as archivemount or fuse-archive for mounting archives with many small files. With it, many metadata I/O accesses to a global file system such as Lustre can be avoided, so that the speed of applications accessing many small files can be improved.

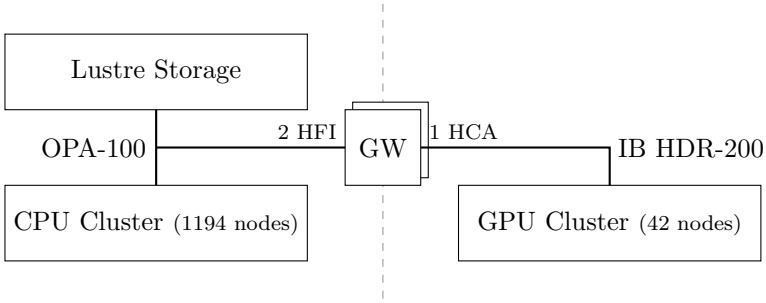

Figure 4: Schematic Overview of the considered components of NHR@ZIB's Lise system using two gateway nodes with four Omni-Path-100 HFIs and two InfiniBand HDR HCAs.

4 ZIB: Lustre in a Heterogeneous Network Environment

ZIB's current NHR system *Lise* comprises a large CPU-only cluster and two system extensions providing GPU resources from Nvidia and Intel. All nodes in the Lise system share a both a GPFS-based file-system (exported via NFS) for the home directories and a 10 PB Lustre work filesystem based upon a DDN appliance. Thus, a consistent view on the data is provided to the users on every node of the system. While the nodes in the CPU-only part of the system are connected via 100 Gbps Omni-Path, the GPU nodes — added during the lifetime of the CPU system — are connected via 200 Gbps HDR InfiniBand, both having a non-blocking fat-tree topology. To enable this connectivity, two gateway nodes, each bridging from one 200 Gbps InfiniBand HCAs to two 100 Gbps Omni-Path HFI, are part of the network infrastructure. Figure 4 provides a schematic overview.

Given the gateways, the theoretical aggregated bandwidth when traversing the network boundaries is 400 Gbps or 50 GByte/s. In contrast, the theoretical limit of the Lustre file system is 175 GByte/s. However, practical measurements in the early lifetime of the Lise system have shown a limit of 76 GByte/s for writing and 91 GByte/s for reading stream performance, respectively.

4.1 Case Study Goal

Using this setup, the question arises if the gateway nodes impose a (practical) bottleneck in the usage of the centralized Lustre storage. To answer this question, the bandwidth of the filesystem will be measured. The GPFS home appliance won't be tested in this study as it is assumed that high-bandwidth workloads will be served by the Lustre filesystem.

Table 1: Performance results for IOR running with (GPU Cluster) and without (CLX Cluster) gateways in the data path. Note that bandwidths are reported in GByte/s, i.e. with a basis of 10.

	Read		Write	
System Cluster	BW (GB/s)	kIOPS	BW (GB/s)	kIOPS
GPU (w/ gateways)	26.9±0.2	3.69±0.03	23.4±0.2	3.2±0.03
CPU (w/o gateways)	99.4±0.9	14.16±0.60	24.7±2.8	3.5±0.07

4.2 Experimental Design

For the measurements, the IOR streaming bandwidth in a file-per-process scenario is measured on two clusters of the system. First, on the Nvdia GPU cluster which consists of 42 nodes and, second, the CPU cluster. On both clusters, the same IOR parameters will be used. However, due to the different CPU architectures on both clusters (2x48C Intel CLX AP vs. 2x36 Intel ICL), the number of processes per node will be set to such the performances on the GPU cluster is maximized. The IOR parameters are experimentally determined to achieve maximum performance on the Nvidia GPU cluster first. After that, the same set of parameters will be applied on the CPU cluster and the obtained performance is compared.

For IOR, 64 tasks where started on each of the 42 GPU cluster nodes using the POSIX backend with direct I/O. The chosen block size was 20 GiB, while the transfer size was set to 8 MiB. To avoid caching effects on the compute node, the artifical memory usage was set to 90% and a random task offset is chosen for IOR between writing and reading. We note that no care was taken to avoid caching effects on the storage system itself. Thus, reading from the Lustre filesystem can be affected by the preceeding write.

4.3 Results

The experiments where conducted using the parameters described above and at the end of a maintenence window in which the complete HPC system was blocked for users. Thus, interference of the benchmark was completely avoided.

The results, i.e. the mean values of three subsequent IOR runs are shown in Table 1 along with the standard deviation. From the data, we observe that write performance for both clusters is very close to each other. While it is lower than what was observed in the early lifetime of the system (see above), it appears to be the limit of what can be achieved with the given setting. We note that no higher bandwidth was observed in the GPU cluster using different parameters. Consequently, we assume that the gateway nodes are saturated.

This appears to be confirmed by the read performance data. While the achieved read bandwidth for the GPU cluser is close to its write bandwidth, the read bandwidth on the CPU cluster is almost 3.6 times higher. This relatively

high bandwidth might be caused by caching on the Lustre storage servers (see above). Nevertheless, while using the same setup, the GPU cluster is not able to achieve similar performance which can be attributed to usage of gateway nodes. Monitoring data also confirmed that the aggregated bandwidth on the gateway nodes was not higher than the value reported above. However, even including the measurement noise, the obtained bandwidth is higher than the theoretical limit for one of the gateways, which confirms both nodes are in use. Nevertheless, they apparently limit the bandwidth for reading which is at the same time well below of the theoretical limit.

4.4 Summary

The case study has shown that gateway nodes can limit the I/O performance. However, the data has also shown that for the given number of nodes, a read bandwidth similar to a direct connection to the storage system can be achieved. While this shows gateway nodes can have an impact on the storage performance, it has to be noted that in practise the I/O performance demand is usually lower than the what has been discussed in this section. Especially for GPU usage by machine learning/AI workloads, it might be interesting to analyse if the performance of meta data operations and operation on small files is affected by the gateway nodes, as latency becomes the dominant component of the performance.

5 RWTH: Applying and monitoring of storage space

The amount of data, and hence fore the amount of required storage space in every scientific discipline is rapidly increasing, especially since the rise of AI researchers tend to keep more and more data. At the same time the number of users on HPC systems is increasing as well. However, storage space on the HPC cluster is often limited and it is not suitable for the long-term storage of files. Therefore, in addition to storage space on the cluster we provide users with storage space on our research data storage. This storage space is envisioned to be used for warm data and is available for at least 10 years.

Applying for storage space on the research data storage system is done by a process that is very similar to applying for computation time. Here we use a specialized JARDS (Joint Application Review and Dispatch Service) instance, that is managed by local research data management experts.

Experience has shown that users often struggle with the amount of data that they want to request. Usually they tend to overprovision by a fair amount. One reason for this is, that it is difficult to foresee how a project will develop over time. Even tough, we allow extending of storage space, application in case more space is required later on, this problem still occurs.

In order to gain a better overview about requested, granted and used quota, we created a detailed reporting on the current state of the storage system in gen-

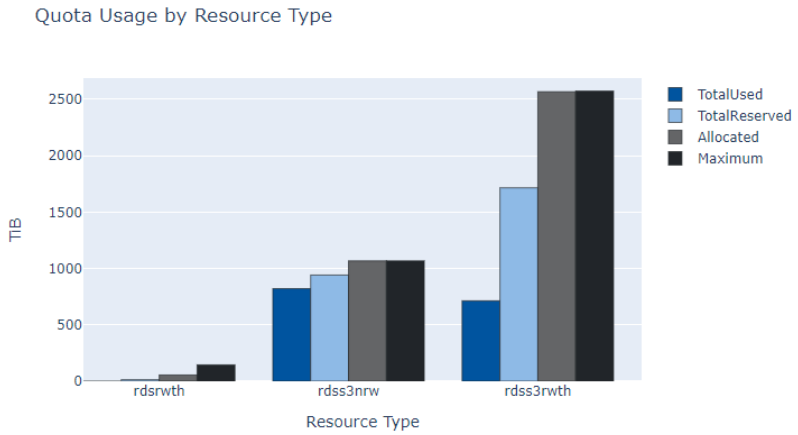

Figure 5: Quota by type

erals as well as the different applications. One part of the reporting is displayed in the figure 5.

The figure shows the amount of applied for, allocated and used quota for the different resource types within our storage system. This has several advantages:

- We can estimate the new amount of data per month and gain an insight in usage of the storage systems.
- We can estimate the average amount of overprovisioning by the researchers
- We gain insight in the amount of overprovisioning that we can do ourselves

When looking at the resource types *rdss3nrw* and *rdss3rwth* you will realize a very different usage pattern. For the *rdss3nrw* the amount of used quota is almost identical to the amount of applied quota. The reason for this is that these are older research project that were migrated to the platform, therefore they are mostly reused and not extended. The *rdss3rwth* resource type on the other hand resembles our actively used resources type, where we receive the most storage space applications. As can be seen in the figure, there is a larger discrepance between applied for, allocated and used quota. As storage space is often applied for well in advance and can be distributed over multiple resources within a research project, it can take some time to allocate the space and some researchers might choose to distribute the remaining quota later on. The difference between allocated and used quota is more interesting. While a single measurement doesn't give much insight, analysing the development of the system enables an estimate of the growth, time until the storage is full and the amount of overprovisioning the operators of the storage could do themselves.

6 Conclusions

With this report, we have shown developments for monitoring and management of storage as well as efficient storage usage for data-intensive HPC. This includes the continuous monitoring of the I/O behavior of compute nodes. The I/O-Weather Map can be used to identify and understand performance flaws and for regression tests to monitor the peak performance of a file system over its lifetime, which might change, e. g. due to configuration changes. Another aspect of storage monitoring is the usage of the file system, for which we discussed the storage space usage by HPC projects compared to their storage request applied for. This is important to better plan capacities, given that research data should be kept for at least 10 years. Regarding storage performance especially for AI applications, we showed that Ratarmount can help to improve speed and reduce load on the parallel file system due to many small files by keeping files in a compressed archive. Ratarmount allows to access these files as if they were unpacked, i. e. no changes in the application are required. Another important aspect of planning procurements is the impact of gateways between different network architectures (Omnipath, Infiniband) on storage performance. The measurements have shown a noticeable impact on the read performance on bandwidth as well as IOPS in this specific setup. In summary, we have shown that getting the best out of HPC storage systems for the various different needs of applications requires tools to monitor the file systems and optimize their performance, which need to be developed further and made available to users and operators.

References

[KB23] Maximilian Knespel and Holger Brunst. Rapidgzip: Parallel Decompression and Seeking in Gzip Files Using Cache Prefetching. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, The 32nd International Symposium on High-Performance Parallel and Distributed Computing, pages 295–307, Orlando, FL, US, June 2023. Association for Computing Machinery.

[KRH+24] Maximilian Knespel, Ashwin Ramaswami, Ryan Hitchman, @RubenKelevra, Corentin Cadiou, Marco Martinelli, Peter Uhrig, rizzel, and Shawn Presser. mxmlnkn/ratarmount: ratarmount, February 2024.