WHITE PAPER

Performance Evaluation of the BeeGFS File System on the Arm AArch64 Architecture

May, 2022

CONTENTS

EXECUTIVE SUMMARY

BeeGFS is one of the leading parallel cluster file systems, developed with a strong focus on performance and designed for easy installation and management.

The Arm architecture has made its way from mobile processors to become a strong alternative to the traditional x86_64 architecture in the desktop and server computing market.

Performance results show that BeeGFS is able to fully utilize the hardware and to saturate the network on a setup with four server machines and six client machines, all on a 100 Gb/s InfiniBand connection and using Arm CPUs.

With the qperf benchmark, we determine a baseline for the maximum network bandwidth of 48.0 GB/s (12.0 GB/s per storage server). Using the IOzone benchmark tool for evaluating sequential I/O, we reach 47.0 GB/s for read and 45.2 GB/s for write accesses, corresponding to 97.9% and 94.2% of the maximum network throughput. With the IOR benchmark, we also show that we are able to reach the maximum bandwidth capabilities of one storage target (mapped to one NVMe drive) with a low thread count of 4 and 24 for write and read accesses, respectively.

Current Arm AArch64 cores for servers are thus capable of taking full advantage of InfiniBand networks and NVMe drives together with BeeGFS on both client and storage nodes.

ABOUT THINKPARQ

ThinkParQ GmbH strives to create and develop the fastest, most flexible, and most stable solutions for every performance-oriented environment. Established in 2014 as a spinoff from the Fraunhofer Center for High-Performance Computing, ThinkParQ drives the research and development of BeeGFS, and works closely with system integrators to create and deliver turn-key solutions.

ABOUT HUAWEI

Founded in 1987, Huawei is a leading global provider of information and communications technology (ICT) infrastructure and smart devices. Huawei has approximately 197,000 employees and operates in over 170 countries and regions, serving more than three billion people around the world. Huawei is a private company wholly owned by its employees.

Huawei's mission is to bring digital to every person, home and organization for a fully connected, intelligent world. Huawei strives for creating value for customers, ensuring secure and stable network operations, promoting industry development through joint innovation with customers and partners, and enabling sustainable development.

INTRODUCTION

BeeGFS is a software-defined storage solution based on the POSIX file system interface, which means applications do not have to be rewritten or modified to take advantage of BeeGFS. A more elaborate overview can be found in [1]. BeeGFS clients accessing the data inside the file system, communicate with the storage servers via network, via any TCP/IP based connection or via RDMA-capable networks, like InfiniBand (IB), Omni-Path (OPA) and RDMA over Converged Ethernet (RoCE). This is similar for the communication between the BeeGFS servers. Furthermore, BeeGFS is a parallel file system. By transparently spreading user data across multiple servers and increasing the number of servers and disks in the system, the capacity and performance of all disks and all servers is aggregated in a single namespace. In this way, the file system performance and capacity can easily be scaled to the level required for the specific use case, also while the system is in production later.

BeeGFS separates metadata from user file chunks on the servers. The file chunks are provided by the storage service and contain the data that users want to store (i.e. the user file contents), whereas the metadata is the "data about data", such as access permissions, file size and the information about how the user file chunks are distributed across the storage servers. A client can talk directly to the storage service to store or retrieve the file chunks as soon as the metadata for a specific file or directory are received. This means, there is no further involvement of the metadata service in read or write operations.

The BeeGFS architecture is composed of four main components:

- Management service: A registry and watchdog for all other services
- Storage service: Stores the distributed user file contents
- Metadata service: Stores access permissions and striping information
- Client module: Mounts the file system to access the stored data

It is possible to run multiple instances with any BeeGFS service on the same machine. These instances can be part of the same BeeGFS file system instance or belong to different file system instances.

The underlying file system in which the BeeGFS services store their data are called management, metadata, or storage targets. While the BeeGFS management and metadata service each use a single target per service instance, the storage service supports one or multiple storage targets for a single storage service instance.

The metadata service is a scale-out service, meaning there can be one or many metadata services in a BeeGFS file system. Each metadata service is responsible for its exclusive fraction of the global namespace, so that having more metadata servers improves the overall system performance. Usually, a metadata target is an ext4 file system at low access latency, e.g., on a flash drive, to improve the responsiveness of the file system.

Similar to the metadata service, the storage service is based on a scale-out design. That means, you can have one or multiple storage services per BeeGFS file system instance, such that each storage service adds more capacity and especially also more performance to the file system. A storage service instance has one or multiple storage targets. The storage service works with any local Linux POSIX file system.

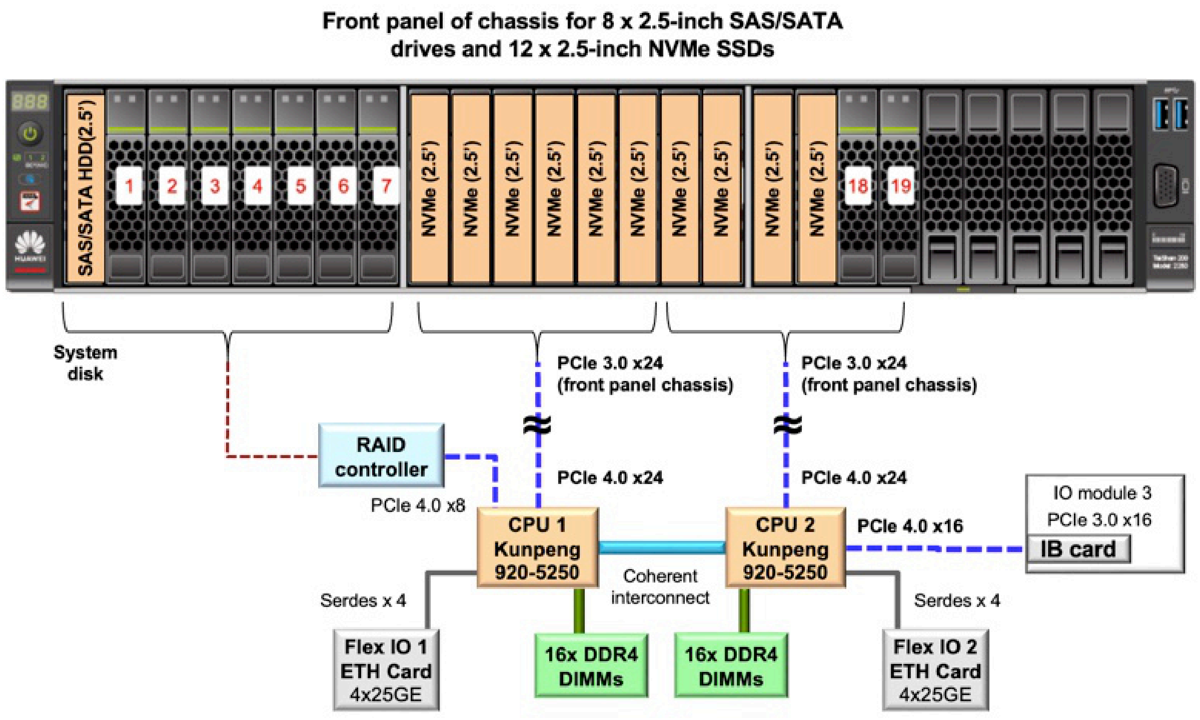
The Arm architecture has made its way from mobile processors to become a strong alternative to the traditional x86_64 architecture in the desktop and server computing market. With its high efficiency and large core counts, it is well suited for highly parallel workloads, making it a suitable choice for compute nodes, metadata and storage servers.

In the following sections, we will evaluate the use of the BeeGFS file system on an Arm AArch64 compute cluster that represents a reasonable installation for R&D labs and small-scale high performance systems. We consider this study as an enablement effort to position BeeGFS on Arm as a forthcoming option to give users more platform choices, as BeeGFS can reach network limits and competitive performance on Arm CPU-enabled systems as well.

EVALUATION SETUP

Hardware Configuration

Server	Huawei TaiShan 200-model 2280
Compute	• HiSilicon Kunpeng 920-5250 CPU: 48 cores at 2.6 GHz clock freq., 48 MB LLC, AArch64 Armv8.2-A core architecture, 150W TDP • Each CPU organized in two NUMA domains (24 cores for each domain), which reflects the chiplet design of the Kunpeng CPU [2] • Direct support for SAS/SATA 3.0 interfaces (up to 16 channels) • Direct support for PCIe 4.0 (up to 40 lanes) • 2 CPUs per node (4 NUMA domains per node)
Memory	• Eight memory channels per CPU (4 channels per NUMA domain) • One 16 GB DDR4-2666 RDIMM per channel • 256 GB total capacity (16 channels with one 16 GB DIMM each)
Storage	Client node: • 1x Huawei ES3000 V5 800 GB SAS SSD system disk Storage node: • 1x Huawei ES3000 V5 800 GB SAS SSD system disk • 10x NVMe SSD Huawei ES3600P V6 with 3.2 TB capacity and PCIe 4.0 x4 connectivity each
Network	Mellanox MT27800 Family ConnectX-5 card, FDR/EDR IB 100 Gb/s, PCIe 3.0 x16
Firmware	CPLD version 5.14, BIOS version 1.38



Organization of a TaiShan 200-model 2280 storage server

Notes

- The chassis type of the selected server that can host up to 12 NVMe drives supports PCIe 3 speeds to the backplane. The CPU itself supports PCIe 4.
- 6 NVMe drives are connected to CPU 1, whereas 4 NVMe drives and the IB NIC are connected to CPU 2.
- The node management network is a separate Ethernet network and not shown.

Topology

10 TaiShan nodes (servers 1 to 10) are available in total, of which 4 are storage nodes (servers 1 to 4). All nodes are connected to the same EDR 36-port non-blocking Mellanox MSB 7700 InfiniBand network switch.

Software Configuration

File system	BeeGFS 7.2.4 using XFS for storage and ext4 for metadata
Operating system	Storage servers 1 to 4: • CentOS 8.4, 4.18.0-305.17.1 Client nodes (servers 5 to 10): • CentOS 8.4, kernel 5.4.173 • Arm PAN feature disabled (Privileged Access Never) • `echo 1 > /sys/block/sda/queue/rq_affinity` • Upgraded from default kernel 4.18 to kernel 5.4 to have full support for queued spinlocks on AArch64 for better performance of high-contended locking
MPI stack	OpenMPI 4.1.1
Standard tests	• Linux *qperf* version 0.4.11 command to characterize network limits between nodes • BeeGFS *StorageBench* to determine maximum performance on storage targets • *MDTest* version 3.4.0 stand-alone test for metadata performance • *IOR* version 3.4.0 stand-alone test of the performance of a parallel file system with sequential read/write accesses of many clients to one file (aka N-1) • *IOzone* version 3.493 and *fio* 3.19 stand-alone tests of the performance of a parallel file system with sequential and random read/write accesses of many clients to many files (aka N-N)

BeeGFS Configuration

We employ servers 1 to 4 for management, metadata and storage, and servers 5 to 10 for client mounts, respectively. The 4 storage servers contain 10 NVMe drives each, as sketched in the above figure. Each NVMe drive is partitioned to provide one storage and one metadata target at a 27:1 capacity ratio, i.e., less than 4% of the capacity is needed for metadata.

Management service

The BeeGFS management service is running on server 1.

Storage services

We allocate one storage service to each NVMe drive and take advantage of the connectivity of drives to NUMA domains of the CPUs. As a result, 3 services are mapped to each of the 2 NUMA domains of CPU 1, whereas 2 services are implemented by each of the 2 NUMA domains of CPU 2. Overall, this leads to 40 storage targets supported by 40 storage services distributed over 4 storage servers.

Metadata services

We use the NUMA domains of CPU 2 (NUMA 2 and 3) to implement 10 metadata services on each of the 4 storage servers, each service responsible for 1 metadata target. Each of the 2 NUMA domains of CPU 2 thus runs 5 metadata services. Overall, this leads to 40 metadata targets supported by 40 metadata services distributed on 4 storage servers. We map metadata services to the NUMA domains of CPU 2 to reduce overall latency since the InfiniBand NIC is connected to CPU 2 as well. In summary, the NUMA mapping of metadata and storage services for all 4 storage servers looks like this:

Server 1 / 2 / 3 / 4	NUMA 1	stor0, stor1, stor2
	NUMA 2	stor3, stor4, stor5
	NUMA 3	meta0, meta1, meta2, meta6, meta7, stor6, stor7
	NUMA 4	meta3, meta4, meta5, meta8, meta9, stor8, stor9

Client mounts

We employ servers 5 to 10 as client nodes. The BeeGFS client code is patched with one enhancement related to Arm PAN functionality released recently with BeeGFS 7.3.0.

BeeGFS parameters

The following BeeGFS tuning options are used:

- connMaxInternodeNum: set to 128 for metadata and 96 for storage on storage nodes, 96 on client nodes
- stripe –chunksize: set to “1m” (1 MiB[1]) for IOR, IOzone and MDtest runs
- tuneNumWorkers: set to 12 for metadata and storage
- tuneTargetChooser: set to “roundrobin” for the metadata configuration

[1]Binary notation: MiB = 2^{20} Byte = 1024^{2} Byte, GiB = 2^{30} Byte = 1024^{3} Byte

PERFORMANCE EVALUATION

In this section, we assess the storage and metadata performance of the cluster configuration described in the previous section, including sequential and random I/O tests.

In the following, we specify throughput and bandwidth values in decimal notation[2], whereas capacities and sizes are specified in binary notation[1]. All presented results are averaged over three runs. We access the NVMe drives in direct I/O mode.

Basic network and NVMe drive characteristics

Network performance: We employ qperf to determine the one-way streaming bandwidth between one client and one server node. qperf returns 12.0 GB/s, as expected for an IB 100 Gb/s connection.

NVMe drive performance with fio: Based on PCIe 3 x4 connectivity, fio reports sequential write bandwidth of 3,060 MB/s and 3,573 MB/s for read accesses; random write IOPS reach 300k, and read IOPS 825k.

We verified the sequential write and read bandwidth values of one NVMe drive with BeeGFS StorageBench as well and achieved the same level of peak performance. This degree of performance approaches the theoretical limit of the underlying PCIe 3 x4 connection (4 GB/s) of the server chassis.

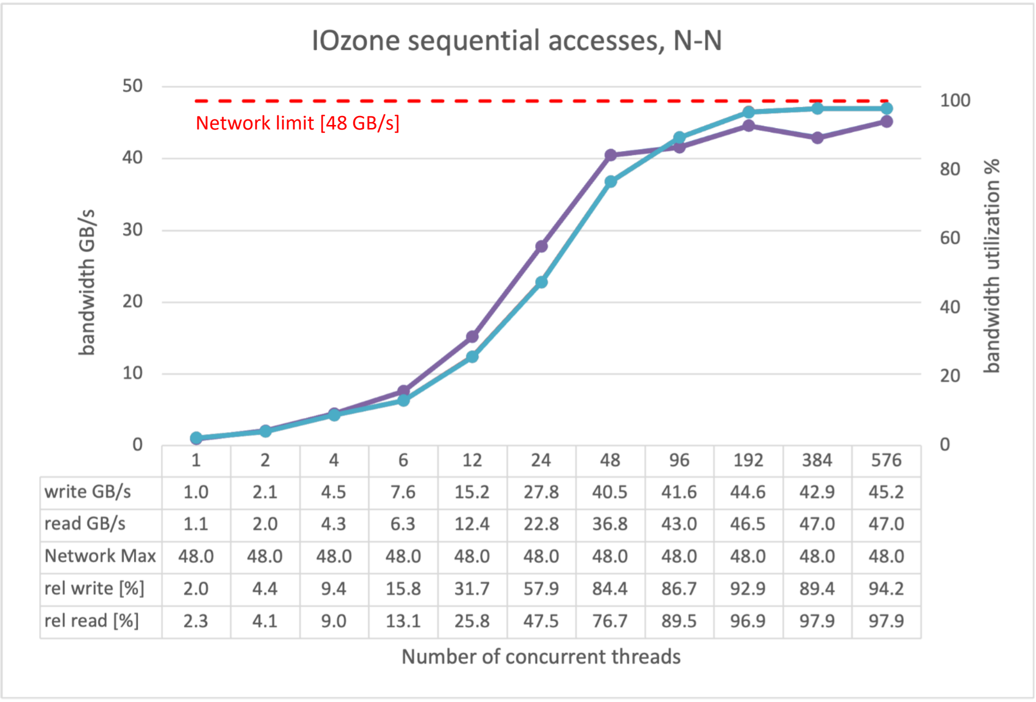
IOzone N-N test, sequential read and write performance

We measure the performance of reads and writes with N threads to N files with IOzone, driven by the 6 client nodes. All 40 storage targets are employed. We vary the number of client tasks as a rational fraction/multiple of the core count of one client node (96 cores per node).

We rely on the following IOzone parameter settings to specify a record size of 1 MiB, a file size of 10 GiB and the use of direct I/O:

```
iozone -i $test -w -c -O -I -r 1m -s 10g
```

The following figure shows the results for sequential accesses.



	1	2	4	6	12	24	48	96	192	384	576
write GB/s	1.0	2.1	4.5	7.6	15.2	27.8	40.5	41.6	44.6	42.9	45.2
read GB/s	1.1	2.0	4.3	6.3	12.4	22.8	36.8	43.0	46.5	47.0	47.0
Network Max	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0
rel write [%]	2.0	4.4	9.4	15.8	31.7	57.9	84.4	86.7	92.9	89.4	94.2
rel read [%]	2.3	4.1	9.0	13.1	25.8	47.5	76.7	89.5	96.9	97.9	97.9

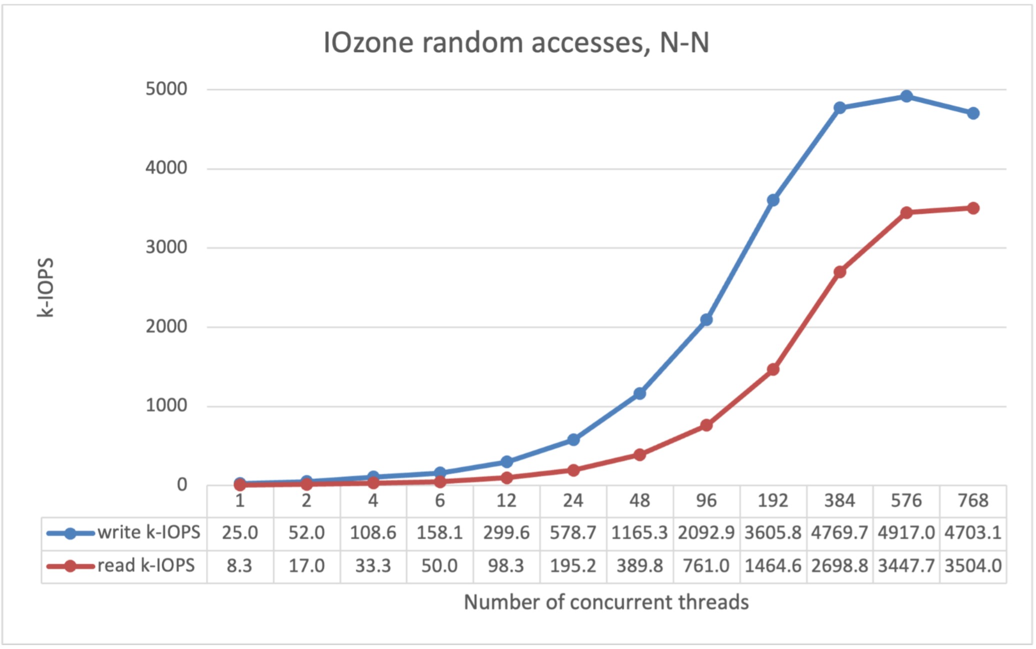
IOzone N-N test, sequential accesses with 40 targets

[2]Decimal notation: MB = 10^6 Byte = 1000^2 Byte, GB = 10^9 Byte = 1000^3 Byte

For the N-to-N test, we see a steady increase in performance up to a task count of 48 where the improvement flattens out, as we come near the limits of the InfiniBand network to the 4 storage servers (4 x 100 Gb/s). Lower task counts benefit from lower latency of posted-write operations over PCIe compared to non-posted reads.

IOzone N-N test, random read and write performance

The following figure shows the results for random accesses characterized by the number of achievable I/O operations using one target. For this test, we reduce the record size to 4 KiB (with the “-r 4k” IOzone option).



	1	2	4	6	12	24	48	96	192	384	576	768
write k-IOPS	25.0	52.0	108.6	158.1	299.6	578.7	1165.3	2092.9	3605.8	4769.7	4917.0	4703.1
read k-IOPS	8.3	17.0	33.3	50.0	98.3	195.2	389.8	761.0	1464.6	2698.8	3447.7	3504.0

IOzone N-N test, random accesses with one target

We can recognize a steady increase in performance up to 384 tasks for random write accesses, after which the improvement flattens out. Using 576 tasks marks the peak for write operations, reaching a score of 4917 k-IOPS, whereas read accesses reach 3504 k-IOPS at 768 tasks. For random accesses, we notice that the read performance saturates below the write performance. The likely reason for this behavior is the lack of further concurrency that can be sustained by the 6 client nodes. Significant higher thread counts would be needed to compensate for the higher round-trip latency for read accesses. The 6 client nodes start to be limited by context switches for 576 threads (6x 96-cores on the client nodes) and beyond.

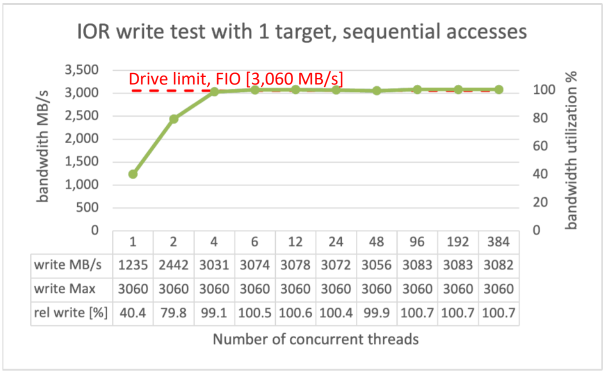
IOR N-1 test, sequential read and write performance

We measure the performance of sequential reads and writes with N threads to a single shared file with IOR, driven by the 6 client nodes.

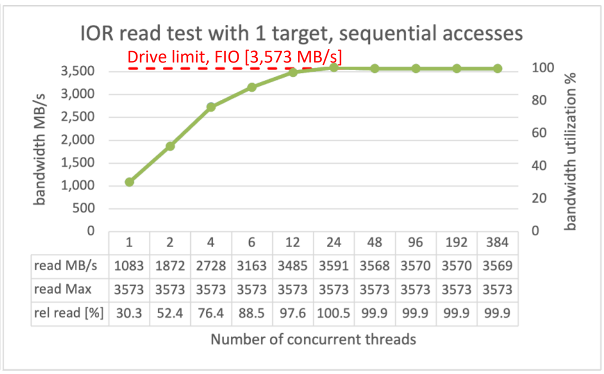
To start with, we only use one storage target to check the feasible performance of one NVMe drive. We vary the number of client tasks as a rational fraction/multiple of the core count of one client node (96 cores per client node). We call the underlying ior command line to specify a block size of 8 MiB and a transfer size of 16 GiB:

```
ior -w -r -i 3 --posix.odirect -t 8m -b 16g -g -d 3 -e -E -o $outfile -s 1
```

.



	1	2	4	6	12	24	48	96	192	384
write MB/s	1235	2442	3031	3074	3078	3072	3056	3083	3083	3082
write Max	3060	3060	3060	3060	3060	3060	3060	3060	3060	3060
rel write [%]	40.4	79.8	99.1	100.5	100.6	100.4	99.9	100.7	100.7	100.7

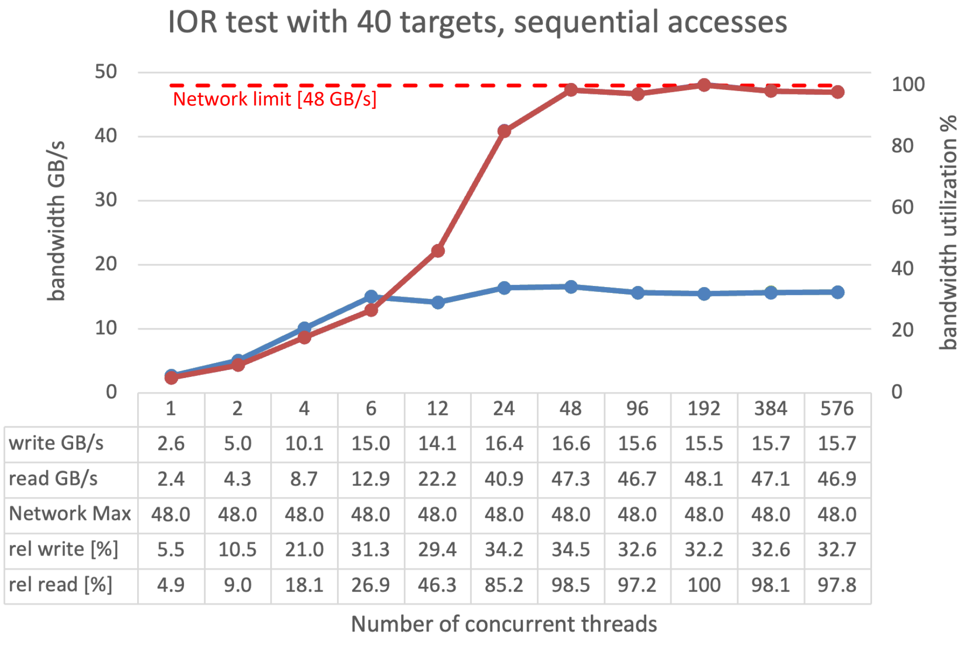


	1	2	4	6	12	24	48	96	192	384
read MB/s	1083	1872	2728	3163	3485	3591	3568	3570	3570	3569
read Max	3573	3573	3573	3573	3573	3573	3573	3573	3573	3573
rel read [%]	30.3	52.4	76.4	88.5	97.6	100.5	99.9	99.9	99.9	99.9

IOR N-1 test, sequential accesses with one target

4 tasks are needed to reach the write performance limit of one target (i.e, one NVMe drive for our configuration) at around 3.1 GB/s, and 24 tasks for read at about 3.6 GB/s. Since read operations on PCIe are non-posted operations requiring a completion packet, more tasks are needed to compensate for the higher latency compared to posted write transactions. Note that the reference baseline was determined with a different test program (results by fio, see subsection on basic characteristics). This is why utilization values slightly above 100% appear in the graph.

We continue with using all available 40 storage targets, using the same underlying ior command as before.



	1	2	4	6	12	24	48	96	192	384	576
write GB/s	2.6	5.0	10.1	15.0	14.1	16.4	16.6	15.6	15.5	15.7	15.7
read GB/s	2.4	4.3	8.7	12.9	22.2	40.9	47.3	46.7	48.1	47.1	46.9
Network Max	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0
rel write [%]	5.5	10.5	21.0	31.3	29.4	34.2	34.5	32.6	32.2	32.6	32.7
rel read [%]	4.9	9.0	18.1	26.9	46.3	85.2	98.5	97.2	100	98.1	97.8

IOR N-1 test, sequential accesses with 40 targets

We see a steady increase in read bandwidth up to 48 tasks. The maximum of 48.1 GB/s is reached for 192 tasks, representing 100% bandwidth utilization of the network to the 4 storage servers. Based on the evaluation of one target, we know that the 40 storage targets could sustain more bandwidth than the InfiniBand network.

As with the single target test, we see that write accesses perform better than read accesses up to 6 tasks, which equals the number of client nodes. The slope of the curve flattens out, and the maximum of 16.6 GB/s for write accesses (meaning 34.5% bandwidth utilization) is measured for 48 tasks.

We attribute the difference between read and write bandwidth for higher task counts to serialization effects in the client with several processes accessing the same inode. Increasing the number of client nodes for this IOR N-1 test would thus be the means to improve bandwidth for write accesses and narrow the gap between read and write performance.

MDtest metadata performance

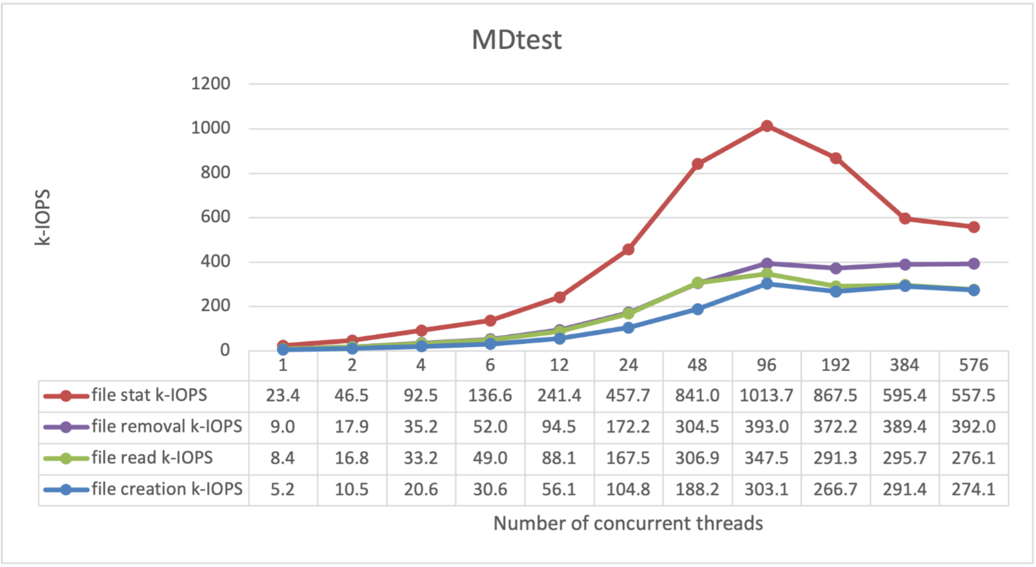
To complement the IOzone results for random accesses and characterize performance in more detail, we stress BeeGFS' metadata performance with MDtest.

We test the performance with 10 metadata targets on each server. We rely on the following underlying command line parameters to fix the number of files per directory to 1024 and again use direct I/O mode:

```
mdtest -i 3 --posix.odirect -b $directories -z 1 -L -I 1024 -y -u -t -F
```

The number of directories is chosen depending on the number of tasks, such that always 3,145,728 files are written (1024 x 384 x 8). In this way, we make sure that a) even with 384 tasks, each task creates a number of directories (8 in this case), and b) the same number of files is processed across tests to require the same number of total IOPS. For 576 tasks, we reduce the total number of files slightly to 2,949,120 (1024 x 576 x 5).

The following figure shows the metadata performance measured by MDtest for 40metadata targets.



	1	2	4	6	12	24	48	96	192	384	576
file stat k-IOPS	23.4	46.5	92.5	136.6	241.4	457.7	841.0	1013.7	867.5	595.4	557.5
file removal k-IOPS	9.0	17.9	35.2	52.0	94.5	172.2	304.5	393.0	372.2	389.4	392.0
file read k-IOPS	8.4	16.8	33.2	49.0	88.1	167.5	306.9	347.5	291.3	295.7	276.1
file creation k-IOPS	5.2	10.5	20.6	30.6	56.1	104.8	188.2	303.1	266.7	291.4	274.1

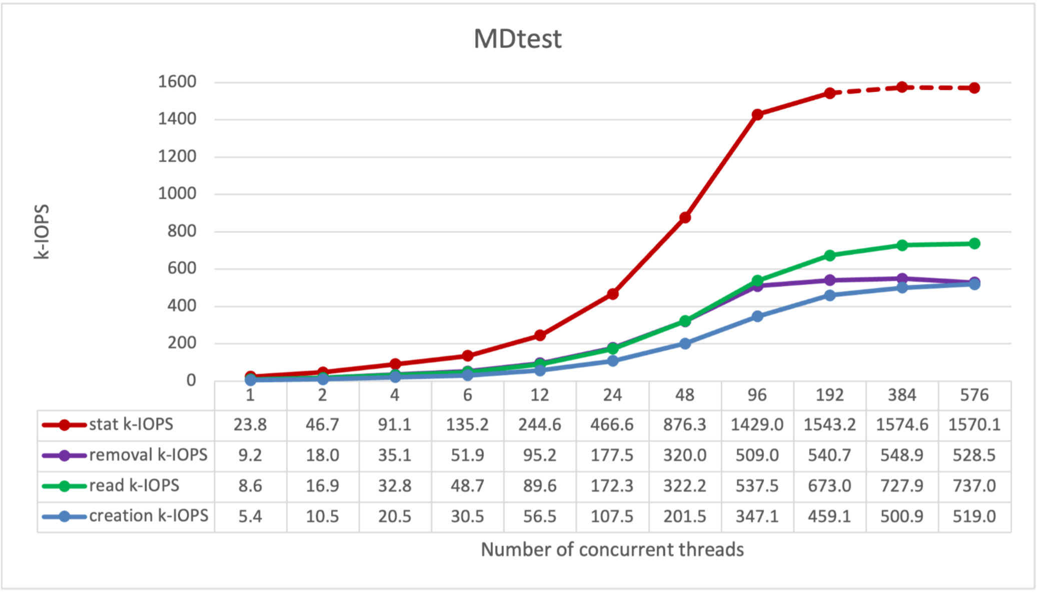
MDtest metadata performance with 40 targets

For MDtest, performance improves steadily up to 96 tasks for all operations. Removal operations stay at the same level of performance beyond 96 tasks, whereas all other operation types drop in performance.

Further optimizing metadata performance for MDtest

In order to reduce access latency to metadata targets further, we only employ the 4 metadata targets of the 4 NVMe drives connected to CPU 2 for each server. As a result, all metadata operations are processed by the CPU that is directly connected to the drives and that has direct access to the NIC for lowest latency.

Compared to the baseline setup, peak removal IOPS increase by about 40%, stat and read IOPS improve by 55% and 112%, and maximum creation IOPS increase by 71%. We do recognize higher variability of results for stat operations at high task counts. However, the shapes of the curves for all operation types appear smoother than for the baseline setup. A more deterministic and lower response time for metadata operations serviced only by CPU 2, coming with a reduction of the traffic between the two CPUs, highlights the performance potential of workload-specific NUMA mappings to reduce latency for communication even on-chip.



	1	2	4	6	12	24	48	96	192	384	576
stat k-IOPS	23.8	46.7	91.1	135.2	244.6	466.6	876.3	1429.0	1543.2	1574.6	1570.1
removal k-IOPS	9.2	18.0	35.1	51.9	95.2	177.5	320.0	509.0	540.7	548.9	528.5
read k-IOPS	8.6	16.9	32.8	48.7	89.6	172.3	322.2	537.5	673.0	727.9	737.0
creation k-IOPS	5.4	10.5	20.5	30.5	56.5	107.5	201.5	347.1	459.1	500.9	519.0

Refined setup: MDtest metadata performance, reduced number of metadata services

Finally, employing only one dedicated NVMe drive per storage server for metadata, thus utilizing 10% of the capacity for metadata, led to lower performance than using the distributed partitions evaluated in this study.

CONCLUDING REMARKS

This White Paper puts emphasis on evaluating BeeGFS on the Arm CPU architecture for the first time. With this initial evaluation we have shown that current Arm AArch64 cores for servers can easily max out the performance of the InfiniBand network and the bandwidth of NVMe drives together with BeeGFS on both client and storage nodes. We also shed light on the mapping of metadata services onto on-chip NUMA domains to take advantage of a modern chiplet-based CPU architecture. If the nodes are augmented with even better NICs (or more NICs) and more client nodes are employed, the presented performance results will scale up further.

BeeGFS 7.3.0 release is fully compatible with PAN, Armv8's "Privileged Access Never" (PAN) feature. Initial tests show no significant change in performance to support this additional layer of security. We switched the PAN feature off for the presented results in this paper.

In this way, we are looking forward to the Arm ecosystem providing complementary solutions for Arm and BeeGFS users in the future.

References

1. Frank Herold, Sven Breuner: An introduction to BeeGFS, ThinkParQ whitepaper, v2.0, 19 pages, June 2018, https://www.beegfs.io/docs/whitepapers/Introduction_to_BeeGFS_by_ThinkParQ.pdf

2. Jing Xia, Chuanning Cheng, Xiping Zhou, Yuxing Hu, Peter Chun: Kunpeng 920: The First 7-nm Chiplet-Based 64-Core ARM SoC for Cloud Services, IEEE Micro, pp. 67-75, Sept./Oct. 2021, https://ieeexplore.ieee.org/abstract/document/9444893

Authors (alphabetical):

Philipp Falk, ThinkParQ GmbH

Matthias Gries, Huawei Technologies Düsseldorf GmbH

Frank Herold, ThinkParQ GmbH

Qinfei Liu, HiSilicon Technologies Co., Ltd.

Maks Marchenko, ThinkParQ GmbH

Troy Patterson, ThinkParQ GmbH

Contact:

info@thinkparq.com