# BeeGFS® Benchmarks on JURON

## Streaming and Metadata Performance on OpenPOWER with NVMe

Alexander Eekhoff, Bastian Tweddell, Dirk Pleiter

October 2017 – v 1.0

## Index

## 1 Overview

**At-A-Glance**

***Juron is a HPC system at FZ Jülich for the Human Brain Project***

JURON is a HPC system delivered by IBM and NVIDIA as part of an R&D contract awarded in the context of a pre-commercial procurement run by the Human Brain Project (HBP). This system is used to demonstrate among others new solutions for integration of dense memory technologies like high-performance SSDs. More specifically, a software stack was developed that allows this memory being accessed through get or put operations within a global address space. In this document, we consider accessing this dense memory via a parallel file system and well-established POSIX semantics.

This document presents a summary of the results of a series of benchmarks executed on BeeGFS services running on the JURON cluster at the Forschungszentrum Jülich. It also presents the best system configuration identified during the experiment.

## 2 System Description

***The cluster consists of 18 POWER8 servers with HGST Ultrastar NVMe SSD***

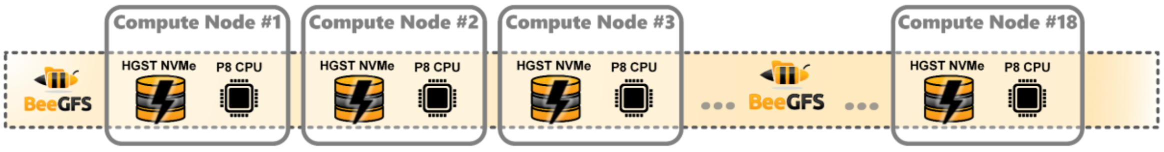

*Figure 1: System Overview*

The cluster consists of 18 nodes. Each node had the following configuration:

- 2 IBM POWER8 processors (up to 4.023 GHz, 2x 10 cores, 8 threads/core)
- 256 GB DDR4 memory
- Non-Volatile Memory: HGST Ultrastar SN100 Series NVMe SSD, partitioned:
  - 1 partition of 745 GB for storage
  - 1 partition of 260 GB for metadata[1]
- Mellanox Technologies MT27700 Family [ConnectX-4], EDR InfiniBand.
  - All nodes were connected to a single EDR InfiniBand switch
- CentOS 7.3, Linux kernel 3.10.0-514.21.1.el7

[1] The system was already prepared with these aligned partitions. The size of the metadata partition could have been much smaller. A single partition for metadata and storage is possible as well.

*JURON was designed as a converged setup*

- During the tests, 2 nodes showed systematically lower performance than the other 16 nodes and were excluded, so that the benchmarks ran on 16 nodes.
- JURON was designed in a converged setup, where computation and storage services run on the same servers. (see Figure 1)

BeeGFS major release 6 was used with the following configuration:

*BeeGFS consists of 3 major services: Storage, Metadata and Client*

- BeeGFS client service version 6.16
- BeeGFS mgmtd, meta, and storage services version 6.14
- The storage and client services were running on all nodes
- For streaming benchmarks, metadata services were running on 2 nodes
  - For streaming benchmarks, metadata performance is not relevant
- For metadata benchmarks, metadata services were running on:
  - (1) 2 nodes
  - (2) All nodes

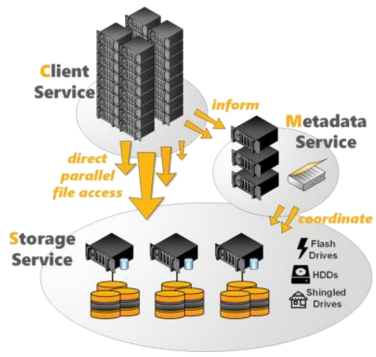

*Figure 2: BeeGFS Services Overview*

## 3 Benchmarks

The following benchmark tools were used to measure performance:

- **IOR 3.0.0** for measuring the sustained throughput of the BeeGFS storage service.
- **Mdtest 1.9.3** for measuring the performance of the BeeGFS metadata service.

# 4 Tuning

The following tables provide the values of the system and service tuning options that led BeeGFS to achieve the highest performance during the experiment. For a detailed explanation of their meaning and how they can be set, please go to https://www.beegfs.io/wiki/TableOfContents.

## 4.1 BeeGFS Storage Service Tuning

***The used tuning options are close to the general storage tuning recommendations on the BeeGFS website: Storage Tuning***

| **Formatting Options** | **Value** |
|---|---|
| NVMe local Linux file system | XFS |
| Partition alignment | yes |

| **XFS Mount Options** | **Value** |
|---|---|
| Last file and directory access | noatime, nodiratime |
| Log buffer tuning | logbufs=8, logbsize=256k |
| Streaming performance optimization | largeio, inode64, swalloc |
| Streaming write throughput | allocsize=131072k |
| Write barriers | nobarrier |

| **IO Scheduler Options**[2] | **Value** |
|---|---|
| Scheduler | deadline |
| Number of schedulable requests (nr_requests) | 1023 (max) |
| Read-ahead data (read_ahead_kb) | 4096 |
| Max kilobytes per filesystem request (max_sectors_kb) | 512 |

| **BeeGFS Storage Service Options** | **Value** |
|---|---|
| Worker threads (tuneNumWorkers) | 64 |
| tuneBindToNumaZone | 0 |

[2] NVMe devices need different scheduler properties compared to HDDs and therefore these values were adapted from the general tuning recommendation on the BeeGFS website.

## 4.2 BeeGFS Metadata Service Tuning

*The used tuning options are close to the general metadata tuning recommendations on the BeeGFS website:* ***Metdata Tuning***

| Formatting Options | Value |
|---|---|
| NVMe local Linux file system | ext4 |
| Minimize access times for large directories | -Odir_index |
| Large inodes | -I 512 |
| Number of inodes | -i 2048 |
| Large journal | -J size=400 |
| Extended attributes | user_xattr |
| Partition alignment | yes |

| EXT4 Mount Options | Value |
|---|---|
| Last File and Directory Access | noatime, nodiratime |
| Write Barriers | nobarrier |

| IO Scheduler Options | Value |
|---|---|
| Scheduler | Same as Storage Service |
| Number of schedulable requests (nr_requests) | Same as Storage Service |
| Read-ahead data (read_ahead_kb) | Same as Storage Service |

| BeeGFS Meta Service Options | Value |
|---|---|
| Worker threads (tuneNumWorkers) | 120 |
| Requests in flight to the same server (connMaxInternodeNum) | 32 |
| tuneBindToNumaZone | - |

## 4.3 BeeGFS Client Tuning

| Options | Value |
|---|---|
| Requests in flight to the same server (connMaxInternodeNum) | 18 |
| Number of available RDMA buffers (connRDMABufNum) | 70 |
| Maximum size of RDMA buffer (connRDMABufSize) | 8192 |
| Remote fsync (tuneRemoteFSync) | true |

## 4.4 BeeGFS Striping Settings Tuning

| Options | Value |
|---|---|
| Chunk size (beegfs-ctl pattern option: --chunksize) | 512K |
| Storage targets per file (beegfs-ctl pattern option: --numtargets) | 1 |

## 5 Results

### 5.1 Sustained Streaming Throughput

For the storage throughput tests, the IOR benchmark was used. At least two times the RAM size of the involved storage servers was used for writing and reading. Each execution started a different number of processes, ranging from 1 to 320, and was repeated 3 times. For less than 16 processes 520 GiB per process and for 16 or more processes a total 8,200 GiB was written and read in each IOR execution. Figure 3 and Table 1 show the mean (± standard deviation) write and read throughput observed in the system.

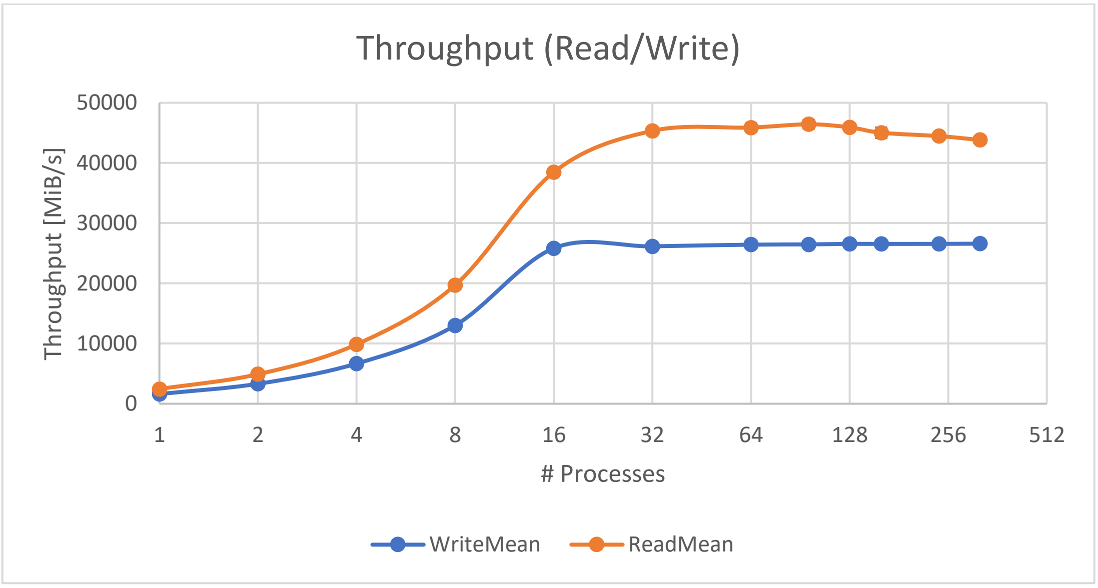

*Figure 3: Read and Write Throughput (Min (#Proc*520GB, 8,200GB); 16 Client Nodes; 16 Storage Targets)*

| | | Write [MiB/s] | | Read [MiB/s] | |
|---|---|---|---|---|---|
| #Processes | #Proc/Node | Mean | StDev | Mean | StDev |
| 1 | 1 | 1612 | 50 | 2442 | 99 |
| 2 | 1 | 3314 | 133 | 4903 | 106 |
| 4 | 1 | 6673 | 155 | 9865 | 102 |
| 8 | 1 | 13022 | 76 | 19716 | 358 |
| 16 | 1 | 25837 | 341 | 38470 | 420 |
| 32 | 2 | 26154 | 64 | 45337 | 560 |
| 64 | 4 | 26427 | 136 | 45890 | 704 |
| 96 | 6 | 26477 | 16 | 46430 | 727 |
| 128 | 8 | 26560 | 94 | 45923 | 343 |
| 160 | 10 | 26553 | 163 | 45019 | 870 |
| 240 | 15 | 26562 | 66 | 44464 | 388 |
| 320 | 20 | 26593 | 129 | 43835 | 274 |

*Table 1: Read and Write Throughput*
*(Min (#Proc*520GB, 8,200GB); 16 Client Nodes; 16 Storage Targets)*

***Measured values are equivalent to 104% (write) an 97% (read) of the NVMe manufacturer specifications***

The results above show that the system can deliver a (mean) maximum of 26,593 MiB/s write and 46,430 MiB/s read throughput. These values are equivalent to 104 % and 97 %, respectively, of the manufacturer specifications of the underlying NVMe devices. The results show approximately linear scaling for both, writing and reading, until saturation when all storages nodes are accessed. For write operations the values are always around 100 % of the manufacturer specifications. For read operations with one process per node the system achieves approximately 80 % of the manufacturer specifications and as soon as two or more processes per node are running, over 90 % of the manufacturer specifications are achieved.

## 5.2 Metadata Throughput

Metadata benchmarks were run with two different configurations. In the first configuration, metadata services were running on two nodes and in the second configuration metadata services were running on all 16 nodes in the used system.

### 5.2.1 Metadata Throughput using 2 Metadata Nodes

Figures 4 and 5 and Table 2 show the mean (± standard deviation) throughput of the metadata service observed in the system when processing a total of 1,000,000 empty files. Each execution was carried out using between 2 to 64 processes and was repeated 3 times.

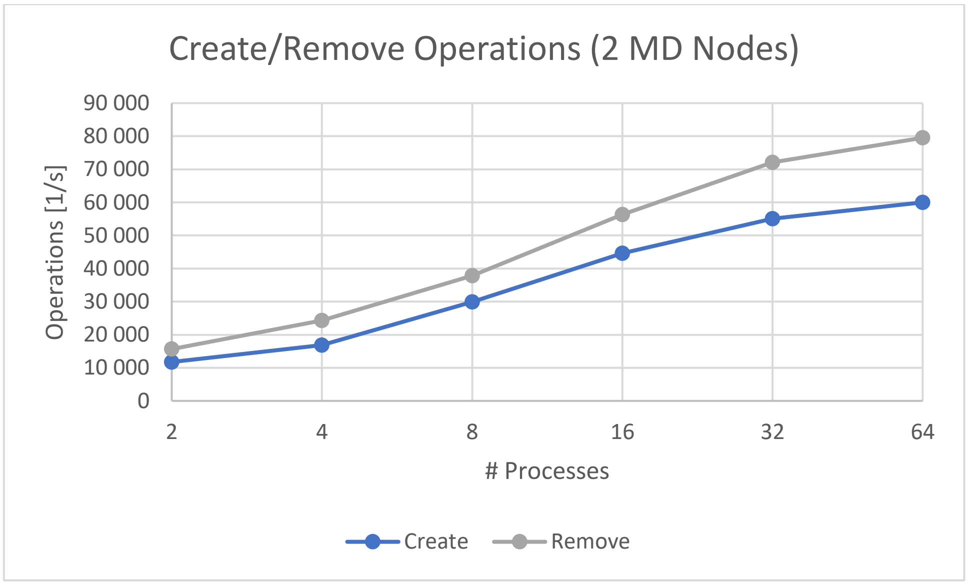

*Figure 4: File Creation and Removal Throughput*
*(1,000,000 Files, 16 Client Nodes, 2 Metadata Services)*

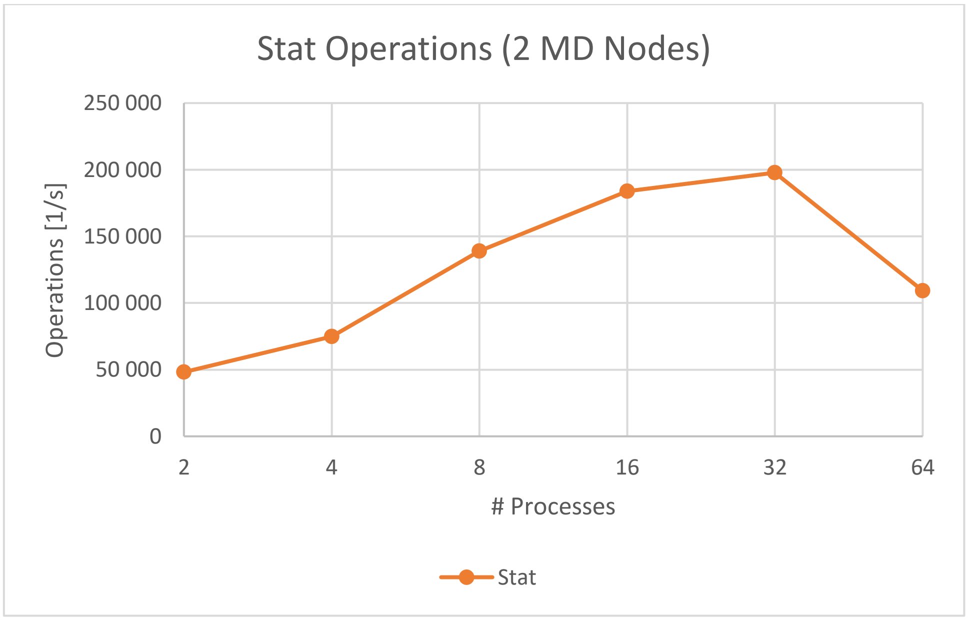

*Figure 5: File Stat Throughput*
*(1,000,000 Files; 16 Client Nodes; 2 Metadata Services)*

| | | Mean [1/s] | | | StDev [1/s] | | |
|---|---|---|---|---|---|---|---|
| #Processes | #Proc/Node | File Creation | File Stat | File Remove | File Creation | File Stat | File Remove |
| 2 | 1 | 11754 | 48161 | 15 670 | 988 | 2835 | 157 |
| 4 | 1 | 16888 | 74821 | 24 329 | 1201 | 4208 | 206 |
| 8 | 1 | 29909 | 138987 | 37 848 | 1033 | 4242 | 239 |
| 16 | 1 | 44664 | 183778 | 56 337 | 462 | 5700 | 212 |
| 32 | 2 | 55089 | 197786 | 72 144 | 210 | 1913 | 798 |
| 64 | 4 | 60009 | 109092 | 79 530 | 794 | 2221 | 426 |

*Table 2: File Creation, Stat and Remove Operations (2 Metadata Services)*

***For file stat operations the throughput is increasing up to 100,000 ops/s per metadata server***

The results above show monotonous scaling for the create and remove operations up to 30,000 operations and 40,000 operations per second and per node, respectively. For the file stat operations, the throughput is also increasing over the main number of processes up to a maximum value of approximately 100,000 operations per second and per node.

## 5.2.2 Metadata Throughput using 16 Metadata Nodes

Figures 6 and 7 and Table 3 show the mean (± standard deviation) throughput of the metadata service observed in the system when processing a total of 8,000,000 empty files. Each execution was carried out using between 4 to 64 processes, and was repeated 3 times.

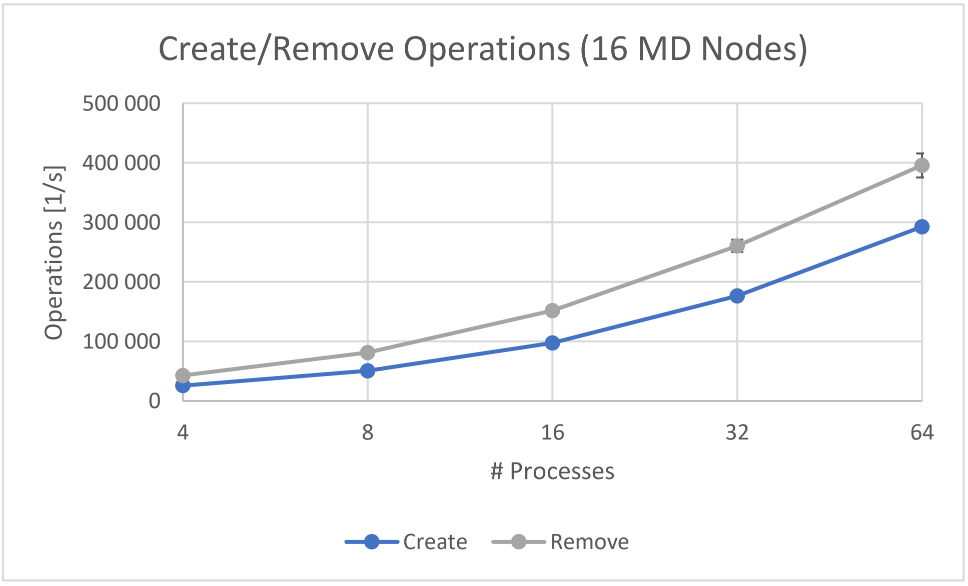

*Figure 6: File Creation and Removal Throughput (8,000,000 Files, 16 Client Nodes, 16 Metadata Services)*

*With 16 metadata servers, the system delivers 300,000 file creates and over 1,000,000 stat operations per second*

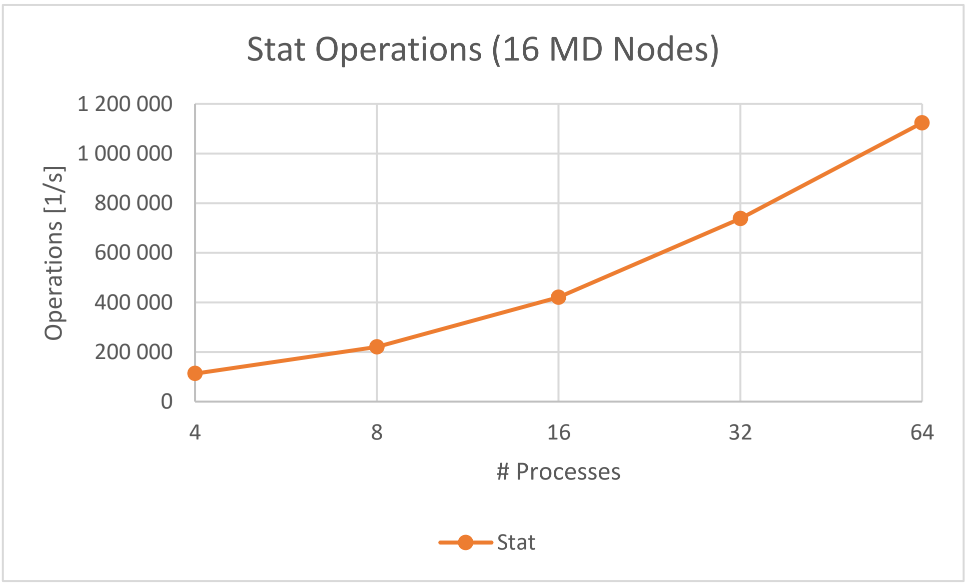

*Figure 7: File Creation and Removing Throughput*
*(8,000,000 Files, 16 Client Nodes, 16 Metadata Services)*

| | | Mean [1/s] | | | StDev [1/s] | | |
|---|---|---|---|---|---|---|---|
| **#Processes** | **#Proc/Node** | **File Creation** | **File Stat** | **File Remove** | **File Creation** | **File Stat** | **File Remove** |
| 4 | 1 | 25768 | 112917 | 42941 | 42 | 186 | 103 |
| 8 | 1 | 50499 | 220310 | 81612 | 39 | 203 | 63 |
| 16 | 1 | 97537 | 420528 | 151805 | 213 | 1274 | 1987 |
| 32 | 2 | 176556 | 737545 | 260537 | 1746 | 13717 | 10344 |
| 64 | 4 | 292357 | 1123413 | 395549 | 888 | 4222 | 20176 |

*Table 3: File Creation, Stat and Remove Operations (16 Metadata Services)*

Like the results with two metadata nodes, the results with 16 metadata nodes show also monotonous scaling for the create and remove operations up to almost 20,000 operations and 25,000 operations per second and per node, respectively. The throughput of file stat operations increases monotonously as well up to a maximum value of approximately 70,000 operations per second and per node.

## 6 Conclusion

BeeGFS on JURON showed excellent benchmark results for storage operations in a converged setup, where applications and BeeGFS storage services are running on the same machines. Based on the high throughput and high number of metadata operations that can be achieved with this system, the internal NVMe drives make this an interesting solution also for burst buffering before staging out computation results to a long-term storage or for prefetching data before a compute job.

The characteristics of the JURON system based on IBM POWER8 processors and NVMe SSDs indicate very good preconditions for compute-intensive and I/O-intensive applications.

## 7 Commands

This section shows the commands that were used on the compute nodes to run the streaming and metadata benchmarks.

### 7.1 IOR

```
#!/bin/bash
ior_dir=~/ior_work
nodes_file=~/nodeslist
max_space=8200
space_per_node=520
num_procs_array=( 1 2 4 8 16 32 64 96 128 160 240 320 )
for num_procs in "${num_procs_array[@]}"; do
    results_file="${ior_dir}/results.log"
    if [ $num_procs -lt 16 ]
    then
        space_per_proc=532480
    else
        space_per_proc=$(($max_space*1024/$num_procs))
    fi
    echo $space_per_proc
    header="processes: ${num_procs} "
    echo ${header}
    printf "\n\n\n${header} \n\n" >> ${results_file}
    mpirun  -hostfile  $nodes_file  --map-by  node  -np  ${num_procs}
~/ior-master/src/ior -wr -i2 -t2m -b ${space_per_proc}m -F -e -g -o
/mnt/beegfs/test.ior | tee -a ${results_file}
done
```

## 7.2 Mdtest

```
#!/bin/bash
mdtest_dir=~/mdtest_work
nodes_file=~/nodeslist
num_files=1000000
num_procs_array=( 2 4 8 16 32 64  )
for num_procs in "${num_procs_array[@]}"; do
    results_file="${mdtest_dir}/results.log"
    files_per_dir=$(($num_files/64/$num_procs))
    header="processes: ${num_procs} "
    echo ${header}
    printf "\n\n\n${header} \n\n" >> ${results_file}
    mpirun  -hostfile  $nodes_file  --map-by  node  -np  ${num_procs}
~/mdtest/mdtest -C -T -d /mnt/beegfs/mdtest -i 1 -I ${files_per_dir}
-z 2 -b 8 -L -F -r -u | tee -a ${results_file}
done
```