Benchmarking Storage Access Patterns – BeeGFS and CephFS

May 7th, 2021

1. Introduction

Parallel File Systems (PFSs) are frequently deployed on leadership High Performance Computing (HPC) systems to ensure efficient I/O, persistent storage, and scalable performance. Storage systems are crucial to the performance of HPC applications [3]. These storage systems interact with node-level filesystems, such as *XFS*[17]*, EXT4* [15]*, BTRFS* [16] etc in different ways causing varying performance impacts on that node [4]. In this project, we aim to present the architectural and system features of BeeGFS and CephFS, and perform experimental evaluations and benchmark the overall performance for various workloads, filesystems, and storage devices.

There have been many efforts in the HPC community to characterize and analyze different performance metrics of various components in supercomputing infrastructure. For instance, one such reference is - "*A Comparative Experimental Study of Parallel File Systems for Large-Scale Data Processing" [1]*. However only a few documentations discuss about performance evaluation for BeeGFS and CephFS. *[2]* compares the performance of the Input/Output load (I/O) of a High-Performance Computing (HPC) application on two different File Systems: CephFS and Lustre. The analysis was performed using a real HPC workload, namely RegCM, a climate simulation application, and IOR, a synthetic benchmark application, to simulate several I/O patterns using different I/O parallel libraries (MPI-IO, HDF5, PnetCDF). And *[3]* presented the architectural and system features of BeeGFS, and performed an experimental evaluation using cutting-edge I/O, Metadata and DL application benchmarks.

Our aim is to implement and assess performance comparisons and evaluate system overheads between two PFSs: BeeGFS and CephFS using multiple file systems – *XFS, EXT4, BTRFS* with different configurations and provide detailed analysis.

2. Motivation

The performance of a Parallel File System depends heavily on its underlying design and deployment. It is critical to characterize the performance and design trade-offs of PFSs with respect to different Filesystems, and their configurations.
With the plethora of available filesystems and their different possible configurations, it is becoming increasingly difficult to choose which underlying filesystem would provide the best performance for a given workload. Adding to that, with latest releases for the File systems, older benchmark results may not be usable. Also with new releases, the storage services become more stable and faster which warrants updated benchmarks. For instance, for CephFS, XFS used to be the recommended underlying filesystem type for production environments, while Btrfs was recommended for non-production environments *[14]*, however with the latest releases these recommendations are no longer applicable.
Also, for Beegfs, it is recommended to use the XFS filesystem for disk partitions of storage targets, because it scales very well for RAID arrays *[13]*. However, there have been significant improvements of ext4 streaming performance in recent Linux kernel versions *[13]*. Hence it becomes significantly useful to maintain an updated benchmark.

3. Background

3.1. CephFS

Ceph is an open-source software (software-defined storage) storage platform, implements object storage on a single distributed computer cluster, and provides 3-in-1 interfaces for object, block and file-level storage. Ceph is developed from the ground up to deliver a single software platform that is self-managing, self-healing and has no single point of failure. Ceph replicates data and makes it fault-tolerant, using commodity hardware , Ethernet IP and requiring no specific hardware support. The Ceph's system offers disaster recovery and data redundancy through techniques such as replication, erasure coding, snapshots and storage cloning. Because of its highly scalable, software defined storage architecture, Ceph is an ideal replacement for legacy storage systems and a powerful storage solution for object and block storage for cloud computing environments.

3.1.1. Key Benefits:

The main advantage of Ceph is that it provides interfaces for multiple storage types within a single cluster, eliminating the need for multiple vendor storage solutions and specialised hardware.
Ceph allows decoupling data from physical hardware storage, using software abstraction layers, providing scaling and fault management capabilities. This makes Ceph ideal for cloud, Openstack, Kubernetes and other microservice and container-based workloads as it can effectively address large data volume storage needs.
Ceph is free and is also an established method, despite its comparably young development history, large amount of helpful information online regarding its set-up and maintenance. In addition, the application has been extensively documented by the manufacturer.
Its scalability and integrated redundancy ensure data security and flexibility within the network. On top of that, availability is also guaranteed by the CRUSH algorithm.

3.1.2. Ceph architecture:

The Ceph storage cluster is made up of several different software daemons. Each of these daemons takes care of unique Ceph functionalities and adds values to its corresponding components:

Monitors – A Ceph Monitor (ceph-mon) maintains maps of the cluster state, including the monitor map, manager map, the OSD map, the MDS map, and the CRUSH map. These maps are critical for cluster state required for Ceph daemons to coordinate with each other. Monitors are also responsible for managing authentication between daemons and clients.

Managers – A Ceph Manager daemon (ceph-mgr) is responsible for keeping track of runtime metrics and the current state of the Ceph cluster, including storage utilization, current performance metrics, and system load. The Ceph Manager daemons also host python-based modules to manage and expose Ceph cluster information, including a web-based Ceph Dashboard and REST API.

Ceph OSDs – A Ceph OSD (object storage daemon, ceph-osd) stores data, handles data replication, recovery, rebalancing, and provides some monitoring information to Ceph Monitors and Managers by checking other Ceph OSD Daemons for a heartbeat.

MDSs – A Ceph Metadata Server (MDS, ceph-mds) stores metadata on behalf of the Ceph File System (i.e., Ceph Block Devices and Ceph Object Storage do not use MDS). Ceph Metadata Servers allow POSIX file system users to execute basic commands (like ls, find, etc.) without placing an enormous burden on the Ceph Storage Cluster.

Ceph client – interfaces read data from and write data to the Red Hat Ceph Storage cluster. Clients need the following data to communicate with the Red Hat Ceph Storage cluster:

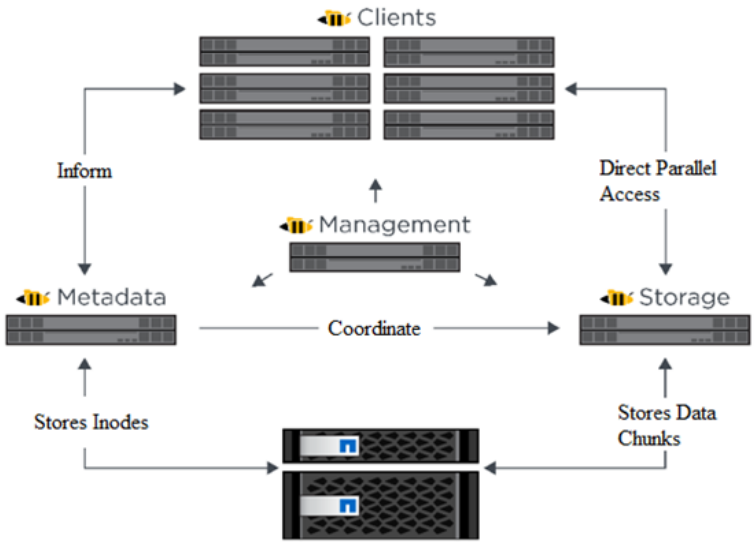
- The Ceph configuration file, or the cluster name (usually ceph) and the monitor address
- The pool name
- The username and the path to the secret key.

Ceph clients maintain object IDs and the pool names where they store the objects. However, they do not need to maintain an object-to-OSD index or communicate with a centralized object index to look up object locations. To store and retrieve data, Ceph clients access a Ceph Monitor and retrieve the latest copy of the Red Hat Ceph Storage cluster map.

3.2. BeeGFS

BeeGFS is a high-performance parallel file system with easy management. The distributed metadata architecture of BeeGFS has been designed to provide the scalability and flexibility that is required to run most demanding HPC applications.

3.2.1. Services:



Client – an internal client registered with the Linux virtual file system interface

Storage – a service to store data chunk files

Metadata – a service to store metadata like directory information and access rights. It can be scaled out to improve the system performance, allowing you to address different types of workloads. Any time a new directory is created, the system automatically selects an available metadata server to handle those files. Sub-directories can be assigned to other servers for purposes of load-balancing.

Management – the conjunction center for metadata, storage, and client services

Admon – a Java-based GUI Administration and Monitoring System Tool

3.2.2. Key Benefits

Distributed File Contents and Metadata

File content is striped across multiple storage servers. Also, the file system metadata is distributed across multiple metadata servers. Large systems and metadata intensive applications in general can greatly profit from the latter feature.

HPC Technologies

Built on highly efficient and scalable multithreaded core components with native Infiniband support, file system nodes can serve Infiniband and Ethernet (or any other TCP-enabled network) connections at the same time and automatically switch to a redundant connection path in case any of them fails.

Easy to Use

BeeGFS requires no kernel patches (the client is a patchless kernel module, the server components are user space daemons), comes with graphical cluster installation tools and allows you to add more clients and servers to the running system whenever you want it.

Optimized for Highly Concurrent Access

BeeGFS is designed to deliver optimal robustness and performance in situations of high I/O load.

Client and Servers on Any Machine

No specific enterprise Linux distribution or other special environment is required to run BeeGFS. BeeGFS clients and servers can even run on the same machine to enable performance increases for small clusters or networks. BeeGFS requires no dedicated file system partition on the servers - It uses existing partitions, formatted with any of the standard Linux file systems, e.g. xfs, ext4 or zfs.

4. Experiment Details

Our objective was to assess the performance comparisons and evaluate system overheads between two selected storage services : BeeGFS and CephFS, we experimented with the following configurations:

- Filesystem – Ext4, Xfs, Btrfs
- Block Size – 1KB, 2KB, 4KB
- Transfer Size – 16MB, 1MB, 64KB
- Access Pattern – Sequential and Random

To log the system activities and network overheads for both the client and server involved in implementing the storage clusters, we utilised the following 2 tools:

SAR : The sar command gathers statistical data about the system. The system maintains a series of system activity counters which record various activities and provide the data that the sar command reports. It extracts the data and saves it, based on the sampling rate and number of samples specified to the sar command.

```
sar -u ALL 1 -o /tmp/cpu > /dev/null 2>&1          # command to record CPU activity
sar -dbp 1 -o /tmp/io > /dev/null 2>&1             # command to record Disk IO activity
sar -n DEV 1 -o /tmp/netdev > /dev/null 2>&1       # command to record Network activity
```

IOR : Is a file system benchmarking application particularly well-suited for evaluating the performance of parallel file systems. IOR can be used for testing performance of parallel file systems using various interfaces and access patterns. IOR uses MPI for process synchronisation – typically there are several IOR processes running in parallel across several nodes in an HPC cluster. As a user-space benchmarking application it is suitable for comparing the performance of different file systems.

Chameleon: We made use of chameleon cloud service to conduct our experiments. Chameleon is an NSF-funded testbed system for Computer Science experimentation. It is designed to be deeply reconfigurable, with a wide variety of capabilities for researching systems, networking, distributed and cluster computing and security. The precise configuration of the workbenches used has been elaborated independently in the respective sections of Ceph and BeeGFS.

Ceph Installation:

There are several different ways to install Ceph. **Ceph-deploy** is a tool for quickly deploying clusters. For the purpose of our experimentation, we used ceph-deploy to install the ceph-cluster with version **v14. 2.12 Nautilus**. Other notable methods to deploy ceph cluster are :

Cephadm - installs and manages a Ceph cluster using containers and systemd, with tight integration with the CLI and dashboard GUI.

ceph-ansible - deploys and manages Ceph clusters using Ansible.

Rook - deploys and manages Ceph clusters running in Kubernetes, while also enabling management of storage resources and provisioning via Kubernetes APIs.

5. Results

The result section is split into 2 main sections due to the differences in the testbed the 2 storage systems were run on.

5.1. Benchmarking CephFS

5.2. Workbench:

For conducting the experiments, the cluster was setup in the following configuration:

Node	CPU	RAM	Extended Hard Drive
Node1	48 Cores	127 GB	None
Node2	48 Cores	127 GB	1.8 TB HDD
Node3	40 Cores	64 GB	1.8 TB HDD
Node4	48 Cores	127 GB	None
Node5	96 Cores	192 GB	None

Table 5.1- Node capabilities

The nodes were connected over a 10-Gigabit Ethernet network. Each node ran Centos 7. Ceph version installed : **v14. 2.12 Nautilus.**

Node#	Node HostName	Ceph Cluster Role
Node1	node1	Admin, Mon
Node2	osd1	Mon, OSD
Node3	osd2	Mon, OSD
Node4	client1	client
Node5	client2	client

Table 5.2 : The ceph services installed on different nodes.

The extended hard drives on the OSD servers: osd1 and osd2, were partitioned to create 50GB partitions. These partitions were allocated to Ceph Cluster as Object Storage. These partitions were mounted with different file systems during different experiments. A Ceph file system requires at least two RADOS pools, one for data and one for metadata. Both the pools were set up with a PG count of 128.

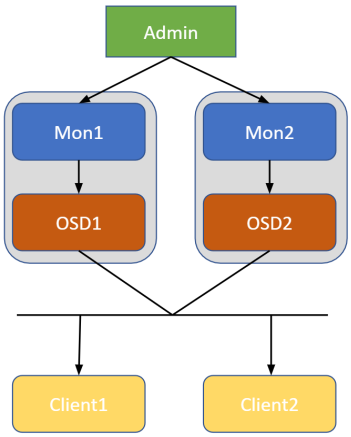


Fig. 5.1 : Ceph workbench architecture overview

5.3. Ceph Results

		16mb				1mb				64kb			
		seq		rand		seq		rand		seq		rand	
		r (MB/s)	w (MB/s)	r (MB/s)	w (MB/s)	r (MB/s)	w (MB/s)	r (MB/s)	w (MB/s)	r (MB/s)	w (MB/s)	r (MB/s)	w (MB/s)
ext4	1kb	141.5	13.88	159.97	12.83	125.66	13.09	135.7	13.15	32.88	3.56	30	3.45
	2kb	181.3	88.56	180.52	93.09	143.15	79.77	160.1	79.56	34.74	16.64	30.63	17.72
	4kb	208.37	99.85	218	102.19	184.03	90.56	189.09	103.65	38.65	26.18	37.81	46.76
xfs	1kb	131.21	16.8	146.02	16.61	121.6	16.83	132.06	15.65	30.17	4.17	26.67	3.26
	2kb	154.44	21.07	158.4	24.51	136.64	18.65	141.81	23.25	33.84	4.85	31.76	7.2
	4kb	160.27	16.6	167.95	18.41	141.83	16.78	138.37	19.88	34.78	5.74	33.6	4.83
btrfs	4kb	174.67	88.03	195.2	97.02	152.74	79.78	156.6	81.69	32.66	19.77	32.97	19.36

Table 5.3

Table 5.3 shows the complete results for the read write speeds for all the file systems and configurations. On the surface we can see that EXT4 with 4KB block size is providing maximum read and write speed. However we can further explore the data on each dimension to analyze more patterns.

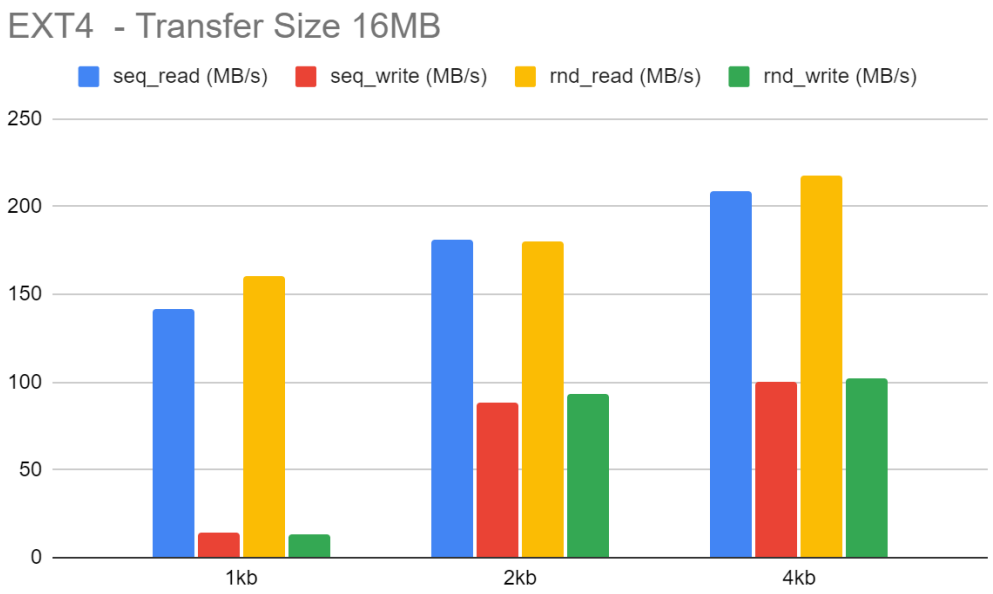


Fig 5.2

Fig 5.2 shows the comparison for read and write speeds when ceph is implemented with EXT4 with varying block size (1KB, 2KB, 4KB), keeping the transfer size of the file fixed as 16MB. We observe there is significant increase in write speeds when changing block size from 1KB to 4KB.

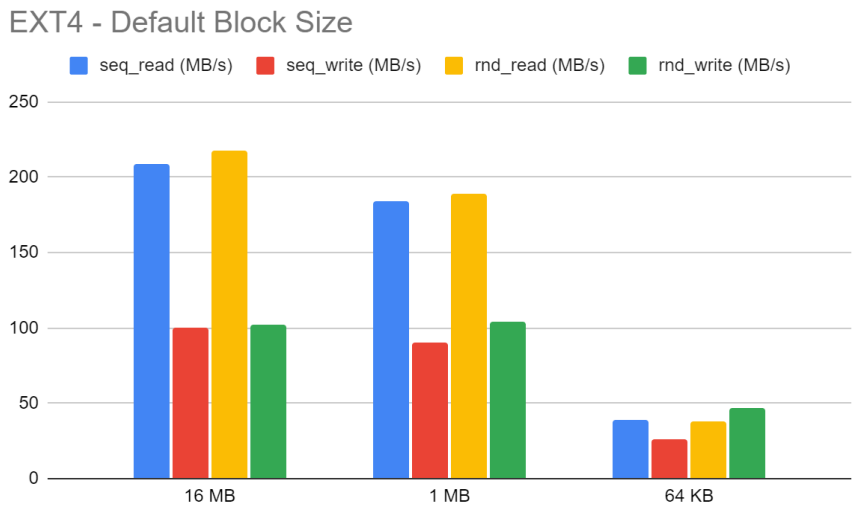


Fig 5.3

Fig. 5.3 presents the comparison between read write speeds for EXT4 file type with default block size, when different file transfer sizes were used for implementing IOR benchmarking. As the transfer size increases the read write bandwidth increases, this is expected as when the block size is less, there would be more Disk I/O operations.

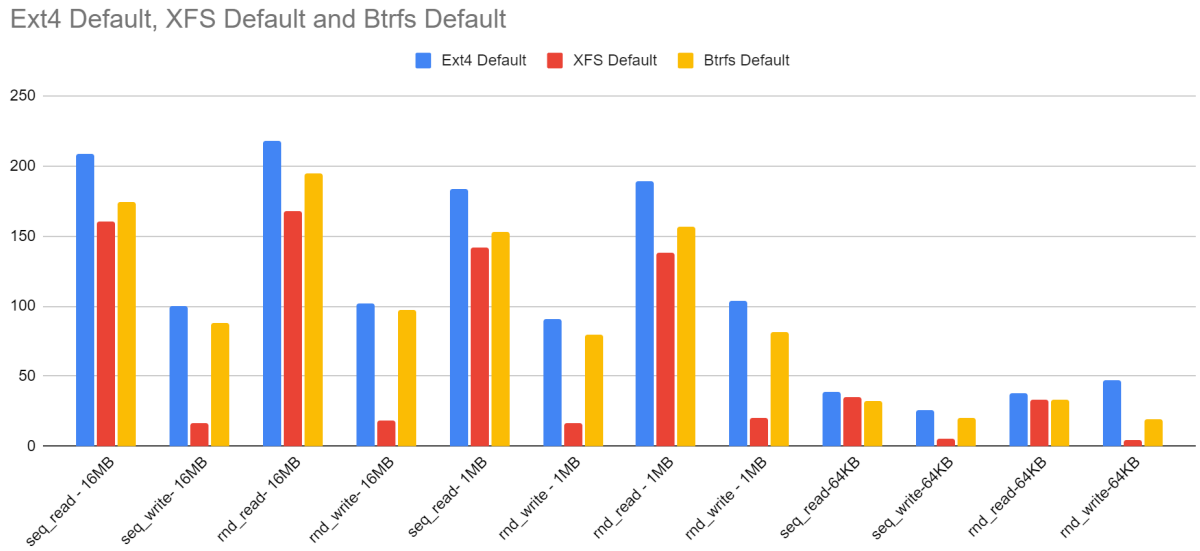
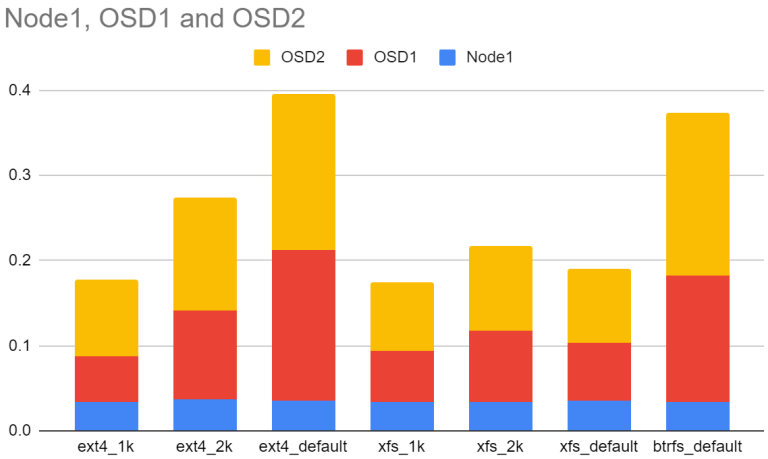


Fig 5.4

From table 5.2 we could see that the default block size for all the file types provided the best performance in terms of read write bandwidth. Therefore in Fig 5.4, we can compare the three file systems in their default block sizes (4KB) with respect to all other configurations. It can be seen that Ext4 shows the best performance. But also we observed that XFS performed poorly for write operations in all the configurations. BTRFS was just slightly lesser in read write performance than EXT4 has mostly outperformed XFS.

5.4. Analysis of Overheads:

Server System Overhead:



The system utilization was furthest away from being the bottleneck for this experiment. Among the three servers for implementing Ceph Cluster - Node1, Osd1, Osd2, none had average CPU utilization more than 0.4%.

Fig. 5.5. System utilization of the three servers

Network Overhead:

Inorder to assess the overhead on the network performance for the Server nodes, we analyzed the following data points from SAR tool:

rd_sec/s : Number of sectors read from the device. The size of a sector is 512 bytes.

wr_sec/s : Number of sectors written to the device. The size of a sector is 512 bytes.

rxkB/s : Total number of kilobytes received per second.

txkB/s : Total number of kilobytes transmitted per second.

The four data points are transformed to the same scale (MB/s) for comparison.

We utilised the SAR tool to extract the transfer and receive statistics of the servers, from this we could infer how much additional data is being transferred than being read/written. In the *Fig.5.6*, *Data received vs data written*, the total data received by the 2 OSDs is compared to the total data written to the disks. The same statistics are represented for the implementation of the Ceph Cluster with the three file systems with default block size. It can be observed that the amount of data received closely follows the data written on the disk, However from *Fig 5.7* We see that while receiving the data by the servers for writing the file to the disk, there is also data transferred from the servers of nearly half the magnitude. This overhead is due to the replication of data maintained by the Ceph Cluster for enabling fault tolerance.

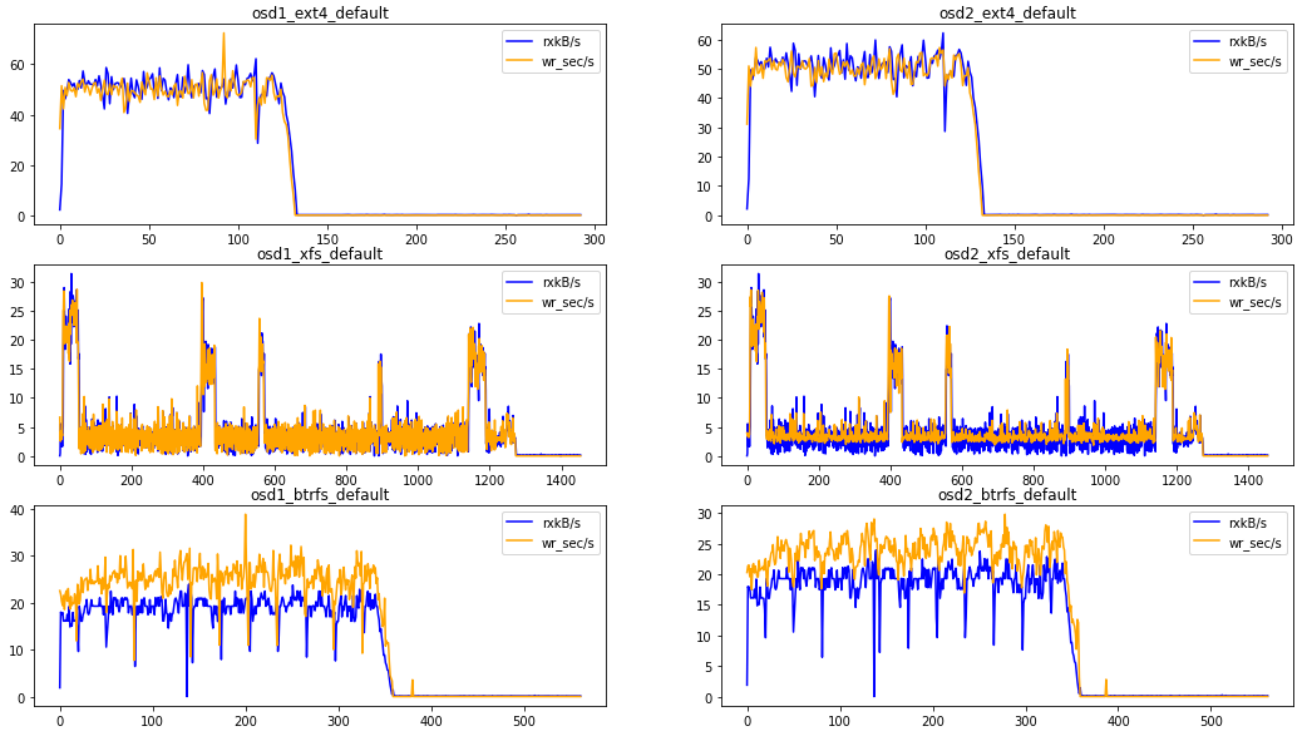


Fig 5.6 Data received vs Data written on disk.

Fig. 5.7 shows Data Transferred vs data read, the two stats closely follow each other and it can be inferred that there is minimum overhead while reading data from the storage by the ceph cluster.

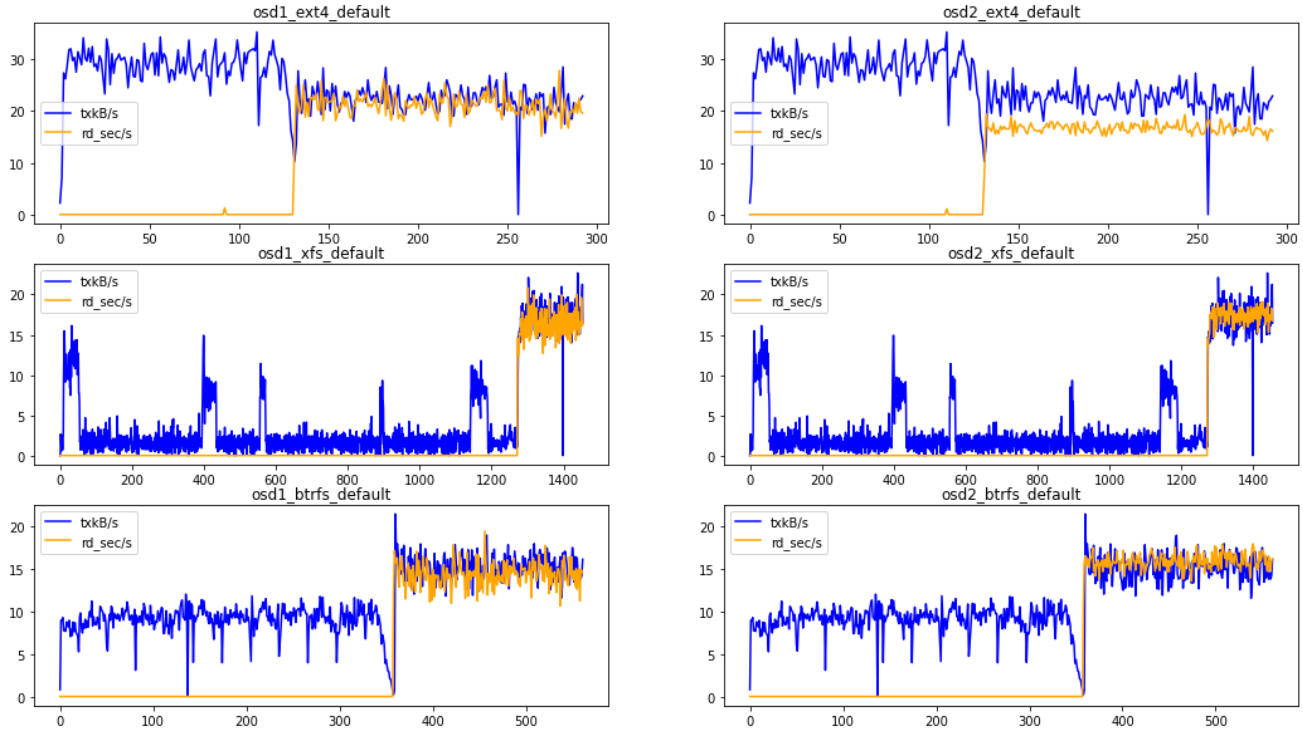


Fig 5.7 Data Transferred vs data read from disk.

Average Disk Utilization:

As seen from the *Fig. 5.8*, for all XFS and BTRFS file systems, the disk utilization follows the same curve. In the first half of the curve, during the write operation, the disk utilization is nearly full ranging from 95% to full, whereas the utilization is saturated for read operations for the given hardware. Also for XFS, the disk utilization is saturated for the entire process. Since for this experiment HDDs were used for storage purposes, it was expected that Disk I/O may become the bottleneck.

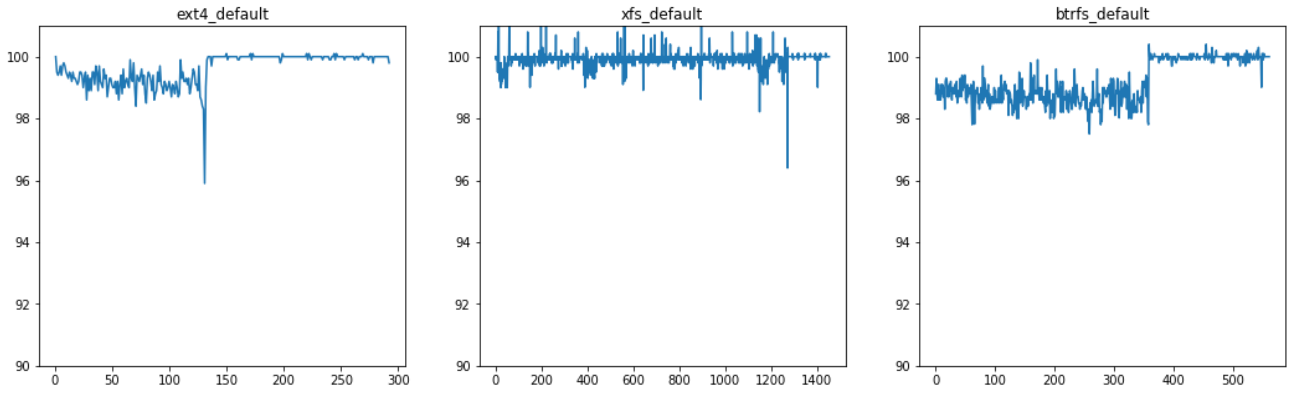


Fig. 5.8 : Disk utilization for the Storage server (OSD1).

5.5. Ceph Findings:

As we compared the three different file systems with varying configurations, we found that EXT4 (default block size) has always performed most optimally for all the transfer sizes we experimented on. BTRFS (default block size) also has been very consistent, performing only slightly less than EXT4 in terms of read write. Although XFS has underperformed compared to the other two file-systems, especially in case of writing in both - Sequential and random access patterns.
CephFS did not show excessive system utilization in implementation with any of the file-systems, leading to the conclusion that Ceph is fairly lightweight in terms of CPU utilization. In terms of overhead while read and write, since CephFS maintains replicated data for enabling fault tolerance, there is an overhead for the network transfer while writing data to the disks. Although this overhead more than compensated by the features Ceph provides such as resilience and Fault Tolerance. Due to which Ceph may also be applicable for working on commodity hardware. Hard Disk utilization has proven to be a bottleneck for this experiment, as for all the file-systems the Disk utilization was nearly saturated for both read and write.

6. Benchmarking BeeGFS:

Workbench

For conducting the experiments, the cluster was setup in the following configuration:

Node	CPU	RAM	HDD
c1	24 Cores	127 GB	250 GB HDD
c2	24 Cores	127 GB	250 GB HDD
s1	48 Cores	127 GB	250 GB HDD
s2	48 Cores	127 GB	250 GB HDD

Table 6.1- Node capabilities

The nodes were connected over a 10-Gigabit Ethernet network. Each node ran Ubuntu 18.04. BeeGFS 7.2.1 was installed using apt package installer.

The BeeGFS services were installed as shown in Table 6.2 below:

Node	BeeGFS Services
c1	Client
c2	Client
s1	Storage, Metadata, Management
s2	Storage, Metadata

Table 6.2 – BeeGFS services per Node

Each server, s1 and s2, were partitioned into root and 3 other partitions of 25 GB each. These 3 partitions were mounted with different file systems during different experiments. The metadata service was given 5 GB of space on each server.

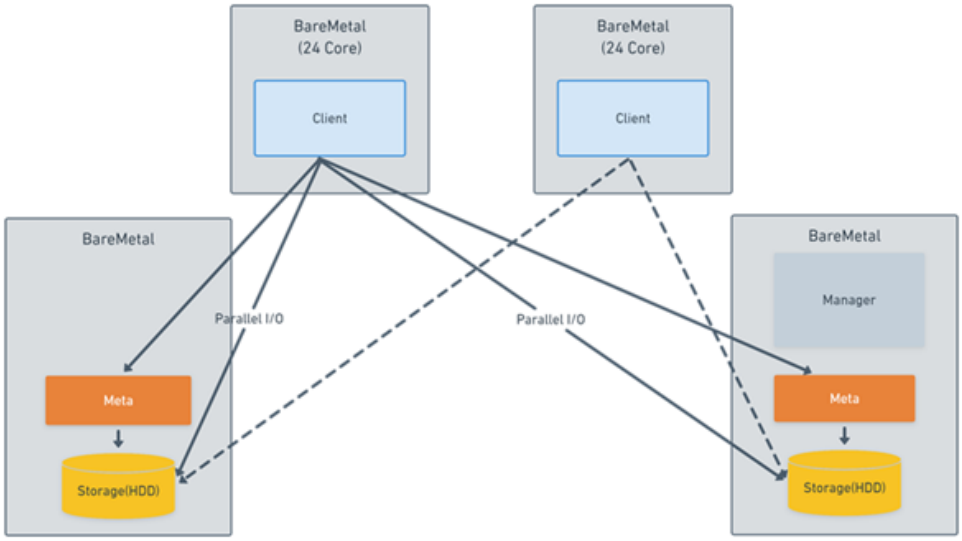


Fig 5.3 - BeeGFS workbench configuration

6.1. Results:

For each of the experiments mentioned in *Section [3]*, 2 clients used 24 threads each and ran IOR benchmarking tool writing and then reading 6 GB of data from the servers. This was repeated for different data sizes over different file systems. During this, the read/write bandwidth and throughput was measured.

Also, the results are divided into 2 sets: bandwidth results and system overhead.

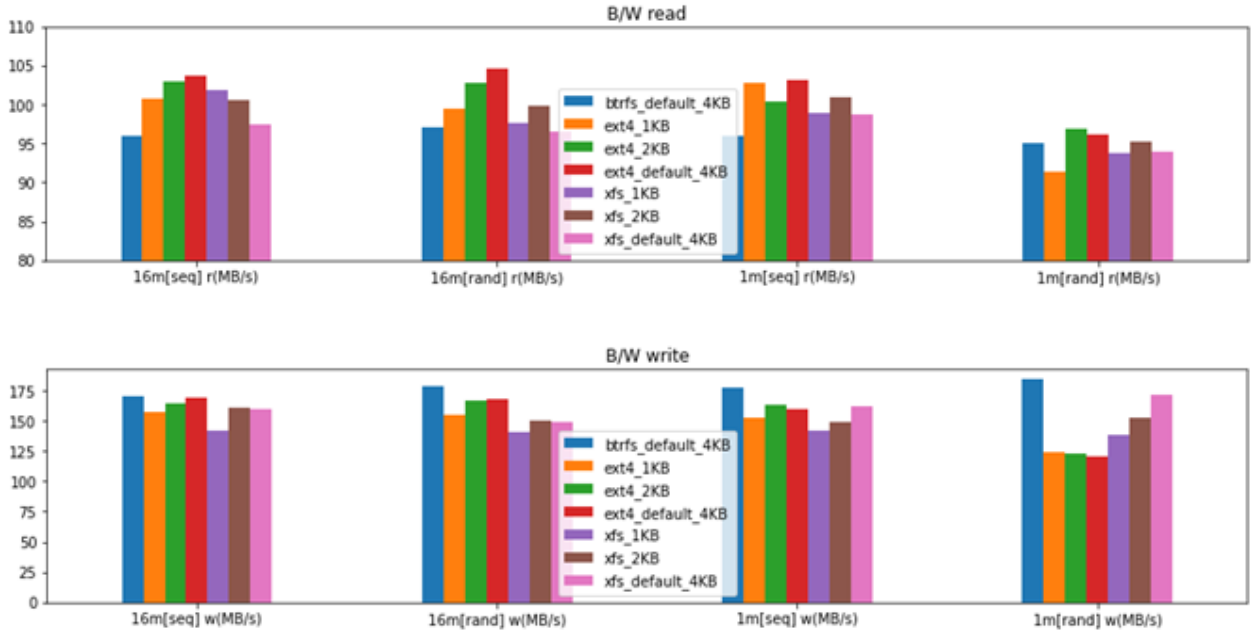
Bandwidth results

The results are divided into large and small workloads for the ease of analysis.

Large workloads

The following table details the read write speeds of 16MB and 1 MB workloads for ex4, xfs and btrfs filesystems for sequential and random reads and writes.

		16mb				**1mb**			
		seq		**rand**		**seq**		**rand**	
		r (MB/s)	**w (MB/s)**	**r (MB/s)**	**w (MB/s)**	**r (MB/s)**	**w (MB/s)**	**r (MB/s)**	**w (MB/s)**
ext4	1kb	100.76	157.14	99.43	154.75	102.82	152.35	91.38	123.77
	2kb	102.87	164.35	102.71	166.31	100.4	163.08	96.84	122.67
	4kb	103.61	168.81	104.63	167.55	103.09	159.41	96.24	120.13
xfs	1kb	101.92	141.59	97.61	140.26	98.84	141.56	93.71	138.66
	2kb	100.49	160.56	99.83	150.69	100.98	149.03	95.16	152.32
	4kb	97.53	159.21	96.46	148.67	98.81	162.26	94.03	171.06
btrfs	4kb	96.01	170.54	97.14	178.86	95.98	177.1	95.06	184



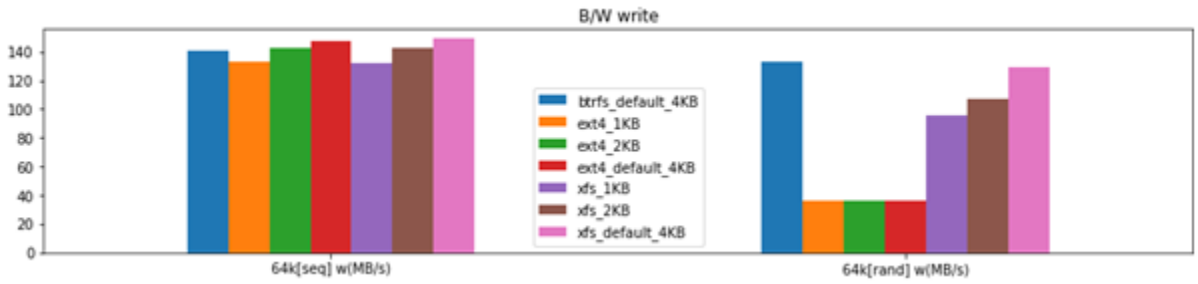
It can be easily seen from the table and graphs above, that,

- for large workloads, ext4 performs best for read operations,
- whereas btrfs performs pretty well for write operations.

Small Workloads

		64kb			
		seq		rand	
		r (MB/s)	w (MB/s)	r (MB/s)	w (MB/s)
	1kb	51.26	132.59	30.37	36.08
	2kb	51.35	142.78	31.12	35.84
ext4	4kb	48.43	146.96	31.62	36.27
	1kb	47.82	132.38	27.84	95.44
	2kb	43.38	142.29	31.96	106.89
xfs	4kb	42.44	148.89	31.06	129.65
btrfs	4kb	41.64	140.54	29.67	132.72

The same results are shown as a form of bar chart for easy readability.



For smaller workloads, the results are quite varied.

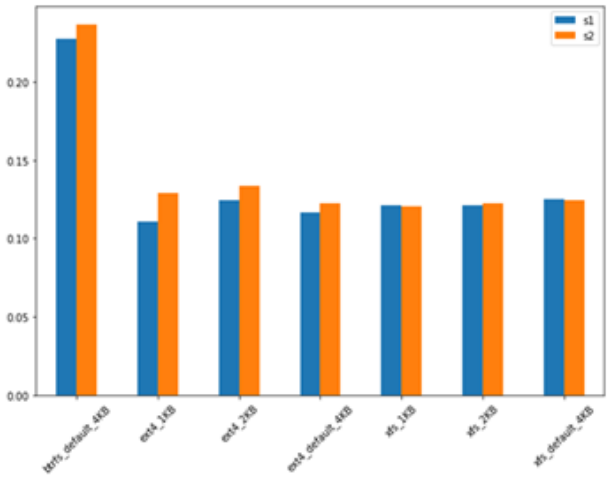
- ext4 performs good for read operations
- btrfs and xfs perform good for write operations

Overhead Results:

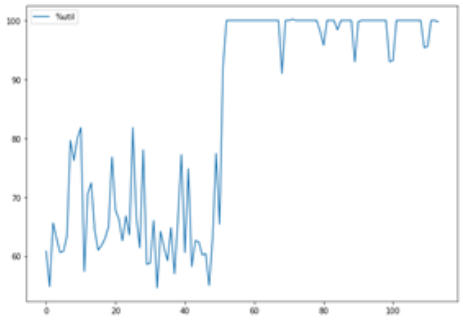
Using SAR utility, CPU, network and Disk I/O statistics were gathered for the experiments on the storage servers s1 and s2.

Average System Utilization

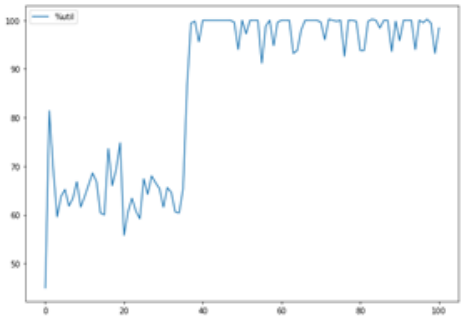
As seen from the graph on the right, the average disk utilization on either of the storage notes does not exceed 0.2% of the total system utilization. We can conclude from this that the BeeGFS services are pretty light weight and can be run on devices with limited compute resources.



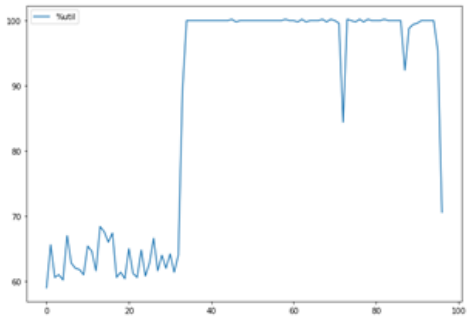
Average Disk Utilization



ext4 (default)



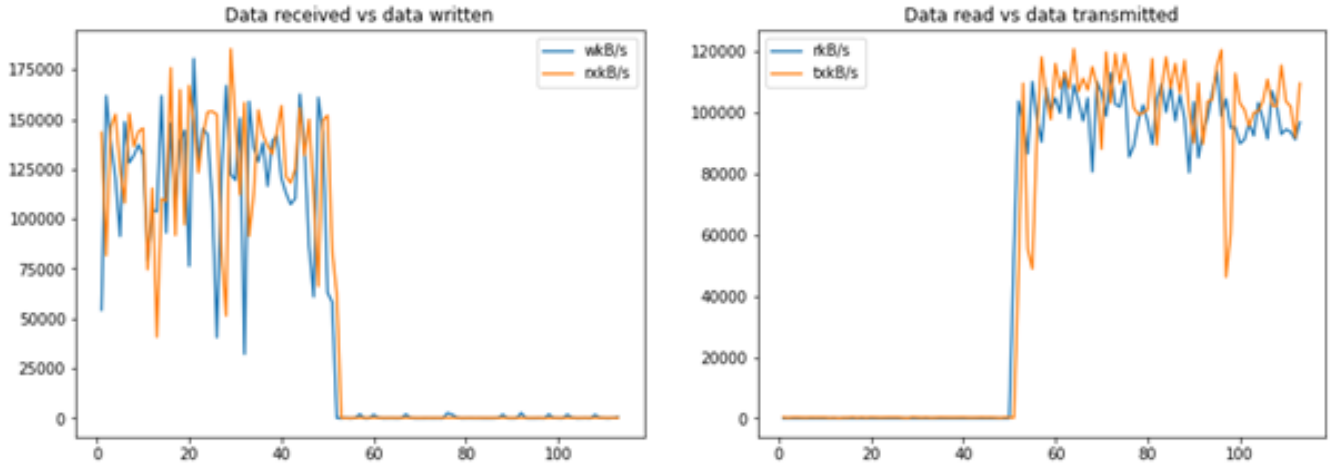
btrfs (default)



xfs (default)

As seen from the graphs, for all different file systems, the disk utilization follows the same curve. In the first half of the curve, during the write operation, the disk utilization varies from 60-80% whereas the utilization is saturated for read operations for the given hardware. This is also evident from the fact that for BeeGFS, the writes are much faster than the reads.

Total Network Overhead (ext4 1MB)



Using network receive and transfer statistics we find out how much additional data is being transferred than being read/written. In the first graph, *Data received vs data written*, the total data received by the 2 storage units is compared to the data total data written to the disks. Whereas, in the second graph, *Data read vs data transmitted*, tells the amount of data actually transferred as compared to the data read from the disk. Here, we can see that the network transfers closely follow the data read/written to the disk. Hence, we can conclude that there is minimal network communication between the nodes and most of the communication is with respect to the actual data required.

6.2. BeeGFS Findings

The 2 sets of findings for BeeGFS are:

6.2.1. Workloads

Given various types of workloads varying from large to small, sequential to random and read heavy to write heavy, following are the findings:

Workload	**FileSystem**
Large Reads	ext4 (4kB)
Large Writes	btrfs (default)
Small reads	ext4 or xfs (default)
Small writes	xfs (default)

We also observe that the read speeds are lower than the write speeds as the HDD saturates in case of reading.

6.2.2. Overheads

The findings for overheads are as follows:

1. Average System utilization < 0.25% even with co-hosting
2. Disk utilization is 100% for read operations
3. Network closely follows the data read/written on to the disk

7. Overall Findings

Following are the overall findings for various *workloads* for the 2 storage services:

Workload	BeeGFS		CephFS	
	FileSystem	**% increase over avg**	**FileSystem**	**% increase over avg**
Large Reads	ext4 (4kB)	2%	ext4 (4kB)	6-9 %
Large Writes	btrfs (4kB)	6-10%	ext4 (4kB) or btrfs (4kb)	4-6 %
Small reads	ext4 (4kB) or xfs (4kB)	10%	ext4 (4kB) or xfs(4kb)	5-10%
Small writes	xfs (4kB) or btrfs (4kB)	5% - 70%	ext4 (4kB)	10-30%

Table 7.1 Performance of best filesystem for varied workloads

Table 7.1 indicates that improvement in performance of the best filesystem over average file speeds of other file systems for different workloads.

With respect to the overall system overhead:

1. The CPU utilization is minimal for either of the storage services
2. For BeeGFS, the disk utilization was saturated in the case of reads whereas for CephFS, it was saturated for both read and writes.
3. For BeeGFS, there was minimal network overhead and the reads and writes closely followed the network transfer rates. Whereas for CephFS, there was an additional overhead observed while writing the data to the disks. This was due to the replication process managed by Ceph for enabling fault tolerance.

8. Conclusion

In this project, we have outlined the functionality of the two Storage services- BeeGFS and CephFS, we successfully completed their installation and setup on Chameleon Cloud. We ran a well-known benchmark - IOR over MPI to measure the bandwidth and latency of the data transfers. We also measured the overhead of the storage services on the system, disk and network using SAR.

We carefully examined the read and write performances of the Storage service with multiple underlying filesystems - *Ext4, Btrfs, Xfs*. For CephFS we were able to conclude that EXT4 with default block size of 4kB provided the most optimal performance on all the configurations that we experimented on. BTRFS was also consistent, as its performances were only slightly lesser than that of EXT4. XFS was not suitable for the workbench set up for our experiment because it underperformed in writing data to storage. In terms of Overhead, the computation requirements for CephFS are fairly minimal. There is some network overhead that CephFS bears to manage replication, which enables Ceph to provide enhanced fault tolerance. Disk utilization proved to be the bottleneck for our experimentation, dictating the overall performance of the system. All the stats for the overhead do not vary much with respect to different underlying file systems.

The performance results for BeeGFS varied for different workloads. For large reads and writes, EXT4 and BTRFS respectively performed best. Whereas, for small reads both EXT4 and XFS were equal and XFS showed the best results for small writes. With respect to the overheads, the CPU utilization was minimal, there was no network overhead and for the disk IO, read operation was the bottleneck for the experiments.

9. Challenges

CephFS setup: Setting up the cluster for Ceph is fairly difficult. Although ceph documentation is well maintained, there are not many posts to help resolve the errors if encountered. The logging for different components and internal services is difficult to access and less readable. Partial knowledge for deploying the Ceph storage architecture can create chaos and also harm the security of the Ceph storage architecture. Their ceph deploy tool does not provide required commands to change/update the OSD. Official documentation for deployment on Ubuntu only supports implementation through MAAS (Metal as a service) and is not well documented for other flavours on other sources.

10. Future works

In future work, we plan to run large-scale experiments to show the scalability of Ceph and BeeGFS in a large-scale cloud. Due to resource constraints, we conducted experiments with the Object Storage devices for the cluster implementation setup using HDDs, this setup can be upgraded by the use of SSDs instead. This can also help incorporate another promising file system - F2FS as part of the experiment.

In this project we solely focused variable configurations of filesystems, however in order to improve the performance of CephFS and BeeGFS we can also further analyse and assess the parameters involved in the setup of the two PFSs.

11. References

[1] *Zoe Sebepou1, Kostas Magoutis, Manolis Marazakis, and Angelos Bilas*, “A Comparative Experimental Study of Parallel File Systems for Large-Scale Data Processing”

[2] *Chiusole, Alberto & Cozzini, Stefano & Ster, Daniel & Lamanna, Massimo & Giuliani, Graziano. (2019).* An I/O Analysis of HPC Workloads on CephFS and Lustre.

[3] *Fahim Chowdhury, Yue Zhu, Todd Heer, Saul Paredes, Adam Moody, Robin Goldstone, Kathryn Mohror, Weikuan Yu,* “I/O Characterization and Performance Evaluation of BeeGFS for Deep Learning”

[4] *BeeGFS Documentation* - https://doc.beegfs.io/latest/

[5] *CephFS Documentation* - https://docs.ceph.com/en/latest/cephfs/index.html

[6] BeeGFS Benchmarks on IBM OpenPOWER P8 Servers - https://www.beegfs.io/docs/whitepapers/IBM_OpenPower_P8_BeeGFS_Performance_by_ThinkParQ.pdf

[7] Ceph Distributed File System Benchmarks on an Openstack Cloud - http://www.cs.utsa.edu/~atc/pub/C51.pdf

[8] IOR Documentation - https://buildmedia.readthedocs.org/media/pdf/ior/latest/ior.pdf

[9] https://phoenixnap.com/kb/linux-create-partition

[10] https://computingforgeeks.com/how-to-deploy-ceph-storage-cluster-on-ubuntu-18-04-lts/

[11] 1.12 Configuring a Block Device on a Ceph Client - https://docs.oracle.com/cd/E37670_01/E66514/html/ceph-ssb_gg1_dt.html

[12] Ceph Ubuntu Documentation - https://ubuntu.com/ceph/what-is-ceph

[13] https://www.beegfs.io/wiki/StorageServerTuning

[14] https://en.wikipedia.org/wiki/Ceph_(software)

[15] https://compas.cs.stonybrook.edu/~nhonarmand/courses/fa14/cse506.2/papers/ols2007v2-pages-21-34.pdf

[16] https://www.researchgate.net/publication/262177144_BTRFS_The_linux_B-tree_filesystem

[17] https://www.usenix.org/system/files/login/articles/140-hellwig.pdf