Rillor

EXPLAINER

Ceph block, object and file storage: which workload uses which

CephFS vs RBD vs RGW, from Ceph's Umbrella 21.2.0 documentation: which workloads need a block device, a shared file system or S3-compatible object storage, and why.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Ceph serves block, file and object storage from one cluster, and the choice between them follows how many clients write the same data, whether the software needs a directory tree and which API it speaks.

- One Ceph cluster serves three interfaces at once: the Ceph Block Device (RBD) for block device images, the Ceph File System (CephFS) for a shared POSIX file system and the Ceph Object Gateway (RGW) for S3- and Swift-compatible object storage, all stored as objects in Ceph's object store, RADOS [3, 4, 9].
- The Ceph Block Device (RBD) lets one client write an image at a time. Its exclusive lock, on by default for new images, makes two clients that write the same image take turns, and its snapshots are crash-consistent unless the guest operating system pauses I/O [5, 6].
- The Ceph File System (CephFS) lets many clients mount one directory tree, with stronger multi-client consistency than NFS. Its Metadata Servers (MDS) handle directories and file attributes, so very large directories, long listings and metadata that outgrows the MDS cache cost time [11, 12].
- The Ceph Object Gateway (RGW) is an HTTP server that offers a large subset of the Amazon S3 API, with versioned buckets, multipart uploads, lifecycle rules and object lock, and uses no Metadata Server. Ceph's documentation points applications that read and write whole files in huge flat collections to it [12, 15, 16, 17].
- In Kubernetes, ceph-csi, Ceph's drivers for the Container Storage Interface (CSI), lists as generally available a Ceph Block Device (RBD) file-system volume for one node, a raw RBD block volume for one node or many, and a Ceph File System (CephFS) volume for many nodes. Kubernetes v1.31 removed the Ceph volume plugins built into Kubernetes itself (in-tree) [21, 23].
- In one pre-release Ceph test on a 10-node, 60-NVMe cluster in March 2023, the Ceph Block Device (RBD) sustained about 25 GB/s of large writes and the Ceph Object Gateway (RGW) about 18.6 GB/s of 4 MB PUTs. The tester traced a big part of the gap to data crossing the network twice through the gateway [29, 30].
- Erasure-coded pools, logical partitions of the cluster that keep each object as data chunks plus coding chunks instead of whole copies, accept the Ceph Object Gateway's (RGW) full-object writes by default. The Ceph Block Device (RBD) and the Ceph File System (CephFS) need overwrites enabled on the pool and keep their metadata in replicated pools [3, 19].
- Ceph's self-selected user surveys put block storage first. In 2018, 84% of 342 respondents used block, 44% file and 42% S3 or Swift [27]. In 2022, of 109 who answered, 91.74% offered the Ceph Block Device (RBD), 58.72% the Ceph File System (CephFS) and 54.13% the Ceph Object Gateway (RGW) [28].

Background

One cluster, three clients

Ceph began as a file system. The 2006 OSDI paper that introduced it stripes file data onto predictably named objects and lets a data distribution function called CRUSH assign those objects to storage devices [32]. Its authors gave metadata a design of its own, because metadata operations make up as much as half of typical file system workloads [32].

Today a Ceph storage cluster is made of four kinds of daemons: Monitors, object storage daemons (OSDs), Managers and Metadata Servers (MDS) [3]. The MDS manages file metadata only when CephFS provides file services [3]. Whatever reaches the cluster, through a block device, object storage, the file system or a custom application written against the librados library, is stored as RADOS objects [3]. RADOS objects live in pools, which are logical partitions, and a pool's size or number of replicas, its CRUSH rule and its number of placement groups decide where objects go [3].

Three clients come with Ceph. RBD provides resizable, thin-provisioned block devices that can be snapshotted and cloned. RGW provides RESTful APIs compatible with Amazon S3 and OpenStack Swift. CephFS provides a POSIX-compliant file system usable with mount or through FUSE in user space [3]. Ceph's file system page calls CephFS the oldest storage interface in Ceph, joined later by RBD and RGW [9]. A 2019 SOSP paper on ten years of Ceph's evolution describes the three as services built on librados and compares RGW with Amazon S3 and RBD with Amazon EBS [31].

Versions and dates

Ceph's releases page lists Umbrella (latest 21.2.0), started 28 October 2026, among the active releases with Tentacle (20.2.5) and Squid (19.2.7), and Reef among the archived releases [1]. Umbrella is the 21st stable release of Ceph [2]. Its notes list a new erasure-coding I/O path, AES-256-GCM server-side encryption in RGW, timestamp-based epochs for versioned objects, a userspace implementation of client-side encryption for CephFS and a multi-threaded CephFS mirror daemon [2].

On the Kubernetes side, the latest release of ceph-csi, Ceph's drivers for the Container Storage Interface (CSI), is v3.18.2, published 2 November 2026 and tested on Kubernetes v1.34, v1.35 and v1.36 [21]. Rook's documentation cited here is for its v1.21 release [24].

Who this report is for

This report is for platform engineers, storage architects and buyers who plan a Ceph cluster for mixed work, such as virtual machines, Kubernetes volumes, datasets and checkpoints, and need to know which interface each workload calls for.

Method

What was examined. For each of Ceph's three interfaces: what the client sees, how many clients can write, where metadata lives, which protocols reach it, how Kubernetes mounts it, what pools and hardware it needs, what published tests measured and what users report running on it.

Sources. Ceph's Umbrella documentation [3-20], Ceph's release pages [1, 2], the ceph-csi repository [21], Kubernetes documentation and its v1.31 release announcement [22, 23], Rook's documentation [24], the Proxmox VE wiki [25, 26], Ceph's 2018 and 2022 user surveys [27, 28, 36], two performance posts on Ceph's blog [29, 30], two peer-reviewed papers [31, 32], Amazon's S3 documentation for the S3 API's write model [33, 34] and PyTorch's distributed checkpoint documentation [35].

How. Each claim comes from the page section, table or passage cited. Web pages were viewed on 8 November 2026, and Ceph's documentation pages carry no page date, so each is cited as Ceph Documentation (Umbrella). Survey percentages are given as the survey reports them, with the number who answered. Nothing was measured for this report, and no figure is computed from the sources.

Reading the S3 sources. Amazon's documents describe Amazon S3. RGW's documentation says it is compatible with the basic data access model and a large subset of the S3 API [15, 16]. This report uses Amazon's pages to describe the API's write model and states no consistency guarantee for RGW itself.

Terms. In Ceph's definition, a block is a sequence of bytes, often 512 [4]. A namespace is the set of names clients use to find data: a directory tree for a file system, buckets and keys for object storage. Kubernetes access modes are ReadWriteOnce (RWO, read-write by a single node), ReadWriteMany (RWX, read-write by many nodes) and ReadOnlyMany (read-only by many nodes) [22].

Findings

1. One cluster, three interfaces

All three interfaces convert their own form of data into RADOS objects and stripe it across many of them [3]. They differ in what sits between the application and the storage daemons.

	RBD	CephFS	RGW
What the client sees	A block device image [3, 4]	A POSIX directory tree [9]	Buckets of S3 or Swift objects [15]
Client	Kernel module or librbd [4]	Kernel mount or FUSE [3, 28]	Any HTTP client using the S3 or Swift API [15]
Data path	Client to storage daemons [4]	Client to storage daemons, with no gateway [9]	Client to the radosgw daemon, then to storage daemons [15, 30]
Metadata	A file system inside the image, unknown to RBD [6]	MDS daemons and a metadata pool [9]	Bucket index and log pools, no MDS [15, 20]

The block device page states that one cluster can serve the gateway, the file system and block devices at the same time [4]. An S3 or Swift object is not the same thing as a RADOS object. One S3 object can map to several RADOS objects [3]. The S3 and Swift APIs share one namespace, and RGW keeps its own users [15].

2. The Ceph Block Device: one writer per image

What it is. Ceph block devices are thin-provisioned and resizable, store data striped over multiple OSDs, and draw on RADOS for snapshots, replication and strong consistency [4]. Clients reach them through kernel modules or the librbd library [4]. The block device page names QEMU, libvirt, OpenStack, OpenNebula and CloudStack as integrations, and its index covers Kubernetes, Nomad, an iSCSI gateway, Windows, an NVMe-oF gateway, mirroring, live migration and encryption [4]. In virtualization, a QEMU/KVM host typically uses librbd to give its guest a block device [3].

One writer at a time. The exclusive-lock feature is enabled on newly created images. It ensures that only one client can write to an image at any given time and is used heavily in virtualization, where it keeps virtual machines from clobbering each other's writes [5]. By default, the lock does not stop two clients from opening one image and writing to it in turns. The lock moves back and forth between them cooperatively and their writes are linearized. The `RBD_LOCK_MODE_EXCLUSIVE` mode, the `--exclusive` option of `rbd device map`, turns off those automatic transitions [5].

Snapshots and clones. A snapshot is a read-only logical copy of an image at a point in time, and snapshot layering clones images, such as virtual machine images, quickly [6]. RBD does not know what file system an image holds, so its snapshots are only crash-consistent unless the operating system that mounts the image coordinates them. Ceph's documentation recommends pausing I/O, running `fsfreeze` or using the QEMU guest agent for virtual machines [6].

On Proxmox VE. The `rbd` storage type is block-level storage with full snapshot and clone functionality. It holds the content types for virtual machine disk images and container root disks, in raw format, as shared storage [25]. Containers always use the kernel RBD client, whatever the `krbd` option says [25].

On Kubernetes. ceph-csi maps RBD images as block devices on worker nodes, optionally mounting a file system inside the image [7]. It uses the RBD kernel modules by default, which may not support every CRUSH tunable or RBD image feature [7].

3. The Ceph File System: a shared POSIX namespace and its metadata path

What it is. CephFS is a POSIX-compliant file system built on RADOS. Ceph names shared home directories, HPC scratch space and shared storage for distributed workflows as its traditional uses [9]. File metadata sits in its own RADOS pool and is served by a resizable cluster of MDS daemons. Clients read and write file data blocks in RADOS directly, and no gateway or broker mediates their data I/O [9].

Why metadata has its own servers. The MDS stores directories, file ownership and access modes. Without it, simple operations such as listing or changing a directory would tax the OSDs unnecessarily, so CephFS keeps metadata in the MDS and file data in cluster objects [3]. Several active MDS daemons split the directory tree into subtrees, and standby daemons take over from a failed one [3].

Consistency. CephFS keeps strong cache coherency across clients, unlike many common network file systems such as NFS [11]. Ceph's page on differences from POSIX lists where it relaxes the standard. With simultaneous writers, a write that crosses object boundaries is not necessarily atomic. A file memory-mapped on several hosts does not propagate writes coherently to other clients' caches. The atime field is not maintained [11]. The page's bottom line is that CephFS relaxes more than local Linux kernel file systems, strictly less than NFS for multi-client consistency and generally less than NFS for write atomicity. Its ordering, from more relaxed to less, is HDFS, NFS, CephFS, then XFS and ext4 [11].

What costs time. Ceph's application guide calls CephFS a network file system, unlike XFS, and highly consistent, unlike NFS [12]. Among the costs it names are these four. A long listing (`ls -l`) lists a directory and then stats every file, which can be slow for large directories. A directory of ten million files works through directory fragmentation but is less efficient than modest-sized directories. Hard links carry a particular performance cost. The MDS acts as a cache for metadata stored in RADOS, and metadata performance is very different for workloads whose metadata fits within that cache [12].

How many MDS daemons. Each file system starts with a single active MDS [13]. Ceph advises adding active daemons when metadata performance is bottlenecked on that one. A single application on a single client typically gains nothing from more daemons unless it runs many metadata operations in parallel, and the workloads that typically benefit are those with many clients, perhaps in many separate directories [13]. A highly available system still needs standby daemons, whatever the number of active ones [13].

On Proxmox VE. The `cephfs` storage type's content types are ISO images, container templates, backup files and snippets. It lists no disk image formats and no clones [26].

4. The Ceph Object Gateway: whole objects over HTTP

What it is. RGW is an object storage interface built on librados that provides a RESTful gateway between applications and the cluster [15]. The radosgw daemon is an HTTP server with its own user management. The S3 and Swift APIs share a common namespace, and object storage does not use the Metadata Server [15]. Ceph states compatibility with a large subset of the Amazon S3 RESTful API and with the basic data access model of the S3 API [15, 16].

S3 features. Ceph's S3 API page marks bucket object versions, multipart uploads, bucket lifecycle, storage classes, bucket notifications and bucket and object policies as supported, and bucket replication as partial, permitted only across zones [16]. With versioning enabled, every object added to a bucket receives a unique version ID [17]. Object Lock places a retention configuration on a bucket, in governance or compliance mode, and works only on a bucket created with object lock enabled [17]. Umbrella added AES-256-GCM server-side encryption and CopyObject for encrypted objects [2].

The S3 write model. Amazon's documents describe the API that RGW is compatible with. A successful PutObject means Amazon S3 added the entire object, never a partial one, and updating one piece of an object's metadata means putting the whole object again [34]. Updates to a single key are atomic, Amazon S3 offers no object locking for concurrent writers, the request with the latest timestamp wins when two PUTs reach one key, and no update is atomic across keys [33]. Amazon also states strong read-after-write consistency for PUT and DELETE requests in Amazon S3 [33]. These are Amazon's statements about Amazon S3, and the Ceph pages reviewed state no equivalent guarantee for RGW.

When Ceph points to it. Ceph's application guide ends by asking whether an application needs a file system at all. It reminds readers that Ceph has an object storage interface and that an application storing huge flat collections of files, reading and writing whole files at once, may be better off on the Object Gateway [12].

What a PUT costs. In the Reef tests, RGW needed three round trips to the primary OSD for each PUT, two synchronous and one asynchronous, to write the data and keep the bucket index synchronized [30].

5. Kubernetes access modes per interface

Access modes. Kubernetes defines ReadWriteOnce as read-write by a single node, ReadOnlyMany as read-only by many nodes and ReadWriteMany as read-write by many nodes, and ReadWriteOncePod, stable since v1.29, as read-write by a single Pod [22]. Access modes do not enforce write protection once storage is mounted, and a volume is mounted with one access mode at a time, even if it supports many [22].

The built-in plugins are gone. Kubernetes v1.31, announced 13 August 2024, removed the CephFS volume plugin and the Ceph RBD volume plugin built into Kubernetes, with the RBD plugin's CSI migration support, which left the `cephfs` and `rbd` volume types non-functional [23]. Both had been deprecated in v1.28, and the release recommends the CephFS and RBD CSI drivers [23]. Kubernetes' persistent volume page still lists the built-in CephFS and RBD plugins in its access-mode table, so this report takes driver support from ceph-csi [21, 22].

What ceph-csi supports. ceph-csi provides independent CSI plugins for RBD and CephFS volumes and is developed and tested only in Kubernetes [21]. Its feature table lists these rows as generally available.

Driver	Volume	Rows listed as generally available	Source
RBD	File mode	ReadWriteOnce volume	[21]
RBD	Block mode	ReadWriteOnce volume, ReadWriteMany volume	[21]
RBD	Either mode	Snapshots, volume from snapshot, volume from another volume	[21]
CephFS	File mode	ReadWriteOnce volume, ReadWriteMany volume, snapshots, volume from snapshot, volume from another volume	[21]

Read together, an RBD volume gives a file system to one node, or a raw block device to one node or many, and CephFS gives a file system to many nodes. Rook's documentation describes a shared file system as one that many pods mount read-write, useful for applications that cluster through a shared file system, and its example container registry claims a ReadWriteMany volume [24]. ceph-csi has no RGW plugin, and applications reach RGW through its REST APIs [15, 21].

6. Erasure coding and pools per interface

Overwrites decide the fit. By default, erasure-coded pools work only with operations that write full RADOS objects, such as RGW's [19]. A per-pool setting, `allow_ec_overwrites`, on BlueStore OSDs only, enables partial writes so that RBD, CephFS and librados can store data in an erasure-coded pool [19]. Even then, RBD

and CephFS keep their metadata in a replicated pool. An RBD image names its erasure-coded pool as its data pool at creation, and CephFS adds one as a data pool at file system creation or through file layouts [19].

The SOSP '19 paper explains the history. Until BlueStore, erasure-coded pools supported only object appends and deletions, so their use was limited to RGW, and RBD and CephFS used only replicated pools [31]. In its RBD overwrite test on a 16-node all-HDD cluster, BlueStore erasure-coded pools reached 6 times the IOPS of FileStore on a 4+2 profile and 8 times on 5+1, which made erasure-coded pools practical for applications that overwrite data, such as RBD and CephFS [31].

Newer optimizations. Since Tentacle, an erasure-coded pool can have optimizations enabled that improve performance for smaller I/Os and remove padding. Ceph says they make erasure coding more suitable for RBD and CephFS and bring little benefit to RGW workloads of large objects read and written sequentially [19]. Umbrella lists a new erasure-coding I/O path among its notable changes [2]. For block and file workloads, Ceph recommends a coding chunk count (m) no larger than 3 [19].

Space arithmetic. An erasure-coded pool's overhead is (k+m)/k, where k is the number of data chunks and m the number of coding chunks, so a 4+2 profile has an overhead of 1.5, against 3.0 for default replication at size 3, which doubles usable space [19]. This is general arithmetic from Ceph's documentation and describes no particular cluster. Ceph adds that CephFS and RGW deployments with many very small files or objects may see considerable extra space amplification on erasure-coded data pools, and that both support several data pools with different media and protection [19].

CephFS pools. A Ceph file system needs at least two RADOS pools, one for data and one for metadata [10]. Ceph recommends at least three replicas for the metadata pool, because losing data there can make the whole file system inaccessible, and low-latency devices such as NVMe or SSDs for it, since they set the latency of client file system operations [10]. Every CephFS inode has at least one object in the default data pool, so Ceph advises a replicated default data pool, with any erasure-coded pool added for chosen directories through file layouts. The default data pool cannot be changed once set [10]. The metadata pool typically holds at most a few gigabytes [10].

What users chose. In Ceph's 2018 survey, 95% of respondents had deployed replication and 25% also used erasure coding [27].

7. Hardware per interface

Ceph's hardware page names different needs for each interface.

- **MDS processors.** CephFS Metadata Servers are CPU-intensive and single-threaded, so they favor CPUs with a high clock rate [20]. The minimum table asks for two cores and prefers higher frequency to more cores [20].
- **MDS memory.** An MDS needs memory in line with its configured cache, with 1 GiB as a minimum for most systems [20].

- **Metadata and index pools.** CephFS metadata pools and RGW index and log pools need SSDs to be effective at enterprise scale, even where hard drives hold bulk data. An RGW deployment with bucket data on hard drives should put every other pool on SSDs [20].
- **Metadata devices.** Ceph's file system guide recommends NVMe or at least SAS or SATA SSDs for the CephFS metadata pool [10].
- **Container images.** Umbrella's container images need x86-64-v3 CPUs [2].

8. Published performance evidence

Two posts on Ceph's blog in March 2023 tested pre-release Reef code on the same 10-node cluster with 60 NVMe drives, at 3x replication [29, 30]. They are the one same-cluster comparison of two Ceph interfaces found for this report.

Test	RBD (librbd)	RGW (S3)	Sources
Large reads	About 71 GB/s	About 53 GB/s for 4 MB GETs	[29, 30]
Large writes	About 25 GB/s (75 GB/s counting replication)	About 18.6 GB/s for 4 MB PUTs	[29, 30]
Small reads	About 4.4 million random read IOPS	About 312,000 4 KB GETs a second	[29, 30]
Small writes	About 800,000 random write IOPS (2.4 million counting replication)	About 178,000 4 KB PUTs a second	[29, 30]

The RGW test used 20 RGW instances and 200 hsbench S3 clients across 512 buckets [30]. Its author explains the large-write gap by the data being transferred twice, once from client to RGW and again from RGW to the OSDs, and concludes that RGW is slightly worse than RBD but not far off once that overhead is counted [30]. RBD's 4 KB synchronous sequential writes averaged under 0.5 ms of latency [29].

No published measurement of CephFS and RBD on the same cluster was found. Ceph's documentation instead names what governs CephFS speed: whether the metadata working set fits in the MDS cache, directory sizes, listings, hard links and the number of active MDS daemons [12, 13], with each MDS single-threaded [20].

9. What users report running on each interface

2018 survey. Ceph's 2018 survey drew 342 completed responses from 52 countries [27]. Of respondents, 84% used block (74% used RBD to boot virtual machines), 44% file and 42% S3 or Swift, and 15% had built native solutions on librados [27]. About 15% re-exported RBD over iSCSI [27]. RGW was used primarily for archive and backup, and 74% chose the S3 API [27]. CephFS workloads were too diverse for a clear leader, the vast majority of clients reached CephFS natively from Linux, with a preference for the in-kernel client over ceph-fuse, 35% made CephFS available to NFS clients and 21% connected clients over CIFS or Samba [27].

2022 survey. The 2022 survey had 127 responses, 105 complete and 22 partial [28], which the accompanying post called a nearly 50% drop in responses year over year [36]. Of 109 who answered, 91.74% offered RBD, 58.72% CephFS and 54.13% RGW [28].

2022 question (answered)	Answers	Source
RBD use cases (98)	Virtualization 73.47%, containers 35.71%, backups 34.69%, archives 17.35%, machine learning 6.12%, HPC 6.12%, OLTP 3.06%	[28]
RBD client access (98)	librbd, for example with QEMU, 78.57%, Linux kernel RBD 58.16%, rbd-nbd 8.16%, iSCSI through tcmu-runner 7.14%, iSCSI through LIO with /dev/rbd 5.10%	[28]
RGW workloads (58)	Backups 68.97%, cloud 43.10%, archives 43.10%, CDN 24.14%, big data and analytics 22.41%, machine learning 8.62%	[28]
CephFS use cases (62)	Home directories 50.00%, backups 45.16%, archives 37.10%, scratch 27.42%, containers 22.58%, HPC 20.97%, big data and analytics 19.35%, machine learning 8.06%	[28]
CephFS access (62)	Linux kernel mount 88.71%, NFS through nfs-ganesha 32.26%, CIFS through Samba 29.03%, ceph-fuse 20.97%, libcephfs 20.97%	[28]

Machine learning here is the survey's answer labelled "AI/ML" [28]. Virtualization leads the RBD answers and home directories the CephFS answers, matching the uses Ceph's documentation names [3, 9, 28]. Backups lead the RGW answers [28]. Respondents chose to take part, so the figures describe them and no wider population.

10. Gateways: NFS, SMB, iSCSI and NVMe-oF

NFS. CephFS namespaces and RGW buckets can be exported over NFS through the NFS-Ganesha server. Ceph's nfs manager module manages the exports, and cephadm or Rook can deploy the NFS-Ganesha daemons [14].

NFS over RGW has limits. RGW namespaces can be exported over NFSv4 alongside S3 and Swift. Buckets appear as directories, and files and directories below them are objects that follow S3 prefix and delimiter conventions [18]. Only full, sequential writes work, so operations such as editing a file in place fail. Links, symlinks included, are not supported, and directories cannot be moved or renamed [18]. An RGW NFS export suits uploads and reads of whole files.

SMB. The surveys record SMB access to CephFS: 21% of 2018 respondents connected clients over CIFS or Samba [27], and 29.03% of 2022 CephFS users used CIFS through Samba [28].

iSCSI and NVMe-oF. The iSCSI gateway exports RBD images as SCSI disks for clients without native Ceph support [8]. It has been in maintenance since November 2022, no longer in active development, and Ceph advises new deployments to reach RBD images directly through librbd, QEMU, Kubernetes CSI and similar clients, or through the NVMe-oF gateway for NVMe-based access [8].

Implications

Matching workloads to interfaces. The findings give a first map from workload to interface.

Workload	Interface	Why	Sources
Virtual machine disks	RBD	One writer per image, librbd in QEMU/KVM, clones from templates	[3, 5, 6]
A Kubernetes volume for one node	RBD in file mode, ReadWriteOnce	Listed as generally available in ceph-csi	[21]
Shared directories that many nodes write	CephFS, ReadWriteMany	One POSIX tree with strong cache coherency	[9, 11, 21]
File-based checkpoints from many ranks	CephFS	PyTorch's file-system writer writes one file per write request into one checkpoint directory, plus a global metadata file when ranks coordinate	[21, 35]
Datasets and weights read and written whole	RGW with versioned buckets	Ceph's advice for flat collections of whole files, a version ID per object	[12, 16, 17]
Clients that speak only NFS or SMB	CephFS through NFS-Ganesha or Samba	RGW's NFS export allows only full, sequential writes	[14, 18, 28]
Block clients without a Ceph driver	NVMe-oF gateway	The iSCSI gateway is in maintenance	[8]

For storage design. Start from writers, namespace and API. A workload with one writer and a file system of its own fits an RBD image, and two writers on one image take turns under the lock [5]. A workload where many clients need the same directory tree fits CephFS, and its design turns on metadata: the number of active MDS daemons, standby daemons, CPU clock rate, MDS cache memory and fast devices for a replicated metadata pool [10, 13, 20]. A workload of whole objects behind an HTTP API fits RGW, with its index and log pools on SSDs [20]. Choose pool protection per interface: erasure coding takes RGW's full-object writes by default, while RBD and CephFS need overwrites and replicated metadata [19].

For platform teams on Kubernetes. Use the ceph-csi drivers, since the built-in plugins stopped working in v1.31 [23]. Pick the volume by access mode: RBD in file mode for one node and CephFS for many [21]. Kubernetes access modes do not enforce write protection after mounting, so on a raw RBD block volume shared by many nodes, Kubernetes leaves write coordination to the software [22]. Check that the kernel RBD client supports the image features in use [7].

For buyers comparing designs. Ask which interface each workload uses and why, how many active and standby MDS daemons a CephFS design has, which pools are replicated and which erasure-coded, and where metadata and index pools sit. Ask whether a performance figure comes from the interface the workload uses. The only same-cluster figures found compare RBD with RGW, on pre-release code from 2023 [29, 30].

Limits of this analysis

- Ceph's documentation is the Umbrella version as viewed on 8 November 2026. Its pages carry no dates, and earlier releases may differ.
- This report measured nothing. The performance figures come from one cluster, one tester and pre-release Reef code in March 2023, at 3x replication [29, 30]. They describe that test and no general ratio between interfaces.
- No published same-cluster measurement of CephFS against RBD was found.
- The user surveys are from 2018 and 2022, with 342 and 127 self-selected respondents [27, 28]. No newer survey with a breakdown by interface was found.
- Amazon's S3 documents describe Amazon S3 [33, 34]. RGW's documentation states compatibility with a large subset of the S3 API and its basic data access model [15, 16], and no RGW consistency guarantee is stated here.
- Only the ceph-csi rows listed as generally available are reported, and the volume metrics rows are left out [21].
- No source reviewed tests PyTorch's distributed checkpoint on CephFS or RGW [35].
- Size limits for RBD images, CephFS files or RGW objects in Umbrella were outside the pages reviewed.

Sources

1. Ceph Releases. Ceph Documentation, Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/latest/releases/

2. Umbrella (v21.2.0 release notes). Ceph Documentation, Ceph Foundation, 28 October 2026, viewed 8 November 2026. https://docs.ceph.com/en/latest/releases/umbrella/

3. Architecture. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/architecture/

4. Ceph Block Device. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/rbd/

5. RBD Exclusive Locks. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/rbd/rbd-exclusive-locks/

6. Snapshots (Ceph Block Device). Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/rbd/rbd-snapshot/

7. Block Devices and Kubernetes. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/rbd/rbd-kubernetes/

8. Ceph iSCSI Gateway. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/rbd/iscsi-overview/

9. Ceph File System. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/cephfs/

10. Create a Ceph file system. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/cephfs/createfs/

11. Differences from POSIX. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/cephfs/posix/

12. Application best practices for distributed file systems. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/cephfs/app-best-practices/

13. Configuring multiple active MDS daemons. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/cephfs/multimds/

14. CephFS and RGW Exports over NFS. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/mgr/nfs/

15. Ceph Object Gateway. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/radosgw/

16. Ceph Object Gateway S3 API. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/radosgw/s3/

17. Ceph Object Gateway S3 API: Bucket operations. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/radosgw/s3/bucketops/

18. NFS (Ceph Object Gateway). Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/radosgw/nfs/

19. Erasure code. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/rados/operations/erasure-code/

20. Hardware Recommendations. Ceph Documentation (Umbrella), Ceph Foundation, viewed 8 November 2026. https://docs.ceph.com/en/umbrella/start/hardware-recommendations/

21. Ceph CSI (README). Ceph project, GitHub repository ceph/ceph-csi, README last changed 6 November 2026, release v3.18.2 published 2 November 2026, viewed 8 November 2026. https://github.com/ceph/ceph-csi

22. Persistent Volumes. Kubernetes documentation, The Kubernetes Authors, last modified 17 June 2026, viewed 8 November 2026. https://kubernetes.io/docs/concepts/storage/persistent-volumes/

23. Kubernetes v1.31: Elli. Kubernetes v1.31 Release Team, Kubernetes blog, 13 August 2024, viewed 8 November 2026. https://kubernetes.io/blog/2024/08/13/kubernetes-v1-31-release/

24. Filesystem Storage Overview. Rook documentation, release v1.21, Rook, viewed 8 November 2026. https://rook.io/docs/rook/latest-release/Storage-Configuration/Shared-Filesystem-CephFS/filesystem-storage/

25. Storage: RBD. Proxmox VE wiki, Proxmox Server Solutions GmbH, last edited 21 May 2026, viewed 8 November 2026. https://pve.proxmox.com/wiki/Storage:_RBD

26. Storage: CephFS. Proxmox VE wiki, Proxmox Server Solutions GmbH, last edited 21 May 2026, viewed 8 November 2026. https://pve.proxmox.com/wiki/Storage:_CephFS

27. Ceph User Survey 2018 results. Ceph blog, ceph.io, 17 July 2018, viewed 8 November 2026. https://ceph.io/en/news/blog/2018/ceph-user-survey-2018-results/

28. Ceph User Survey 2022 results (PDF). Ceph User Survey Working Group, Ceph Foundation, published with the post of 12 July 2022, viewed 8 November 2026. https://ceph.io/en/news/blog/2022/ceph-user-survey-results-2022/images/ceph-user-survey-results-2022.pdf

29. Ceph Reef Freeze Part 1: RBD Performance. Mark Nelson, Ceph blog, 27 March 2023, viewed 8 November 2026. https://ceph.io/en/news/blog/2023/reef-freeze-rbd-performance/

30. Ceph Reef Freeze Part 2: RGW Performance. Mark Nelson, Ceph blog, 30 March 2023, viewed 8 November 2026. https://ceph.io/en/news/blog/2023/reef-freeze-rgw-performance/

31. File Systems Unfit as Distributed Storage Backends: Lessons from 10 Years of Ceph Evolution. Aghayev, Weil, Kuchnik, Nelson, Ganger and Amvrosiadis, ACM SOSP '19, October 2019, Carnegie Mellon Parallel Data Lab copy, viewed 8 November 2026. https://www.pdl.cmu.edu/PDL-FTP/Storage/ceph-exp-sosp19.pdf

32. Ceph: A Scalable, High-Performance Distributed File System. Weil, Brandt, Miller, Long and Maltzahn, 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06), 2006, viewed 8 November 2026. https://www.usenix.org/legacy/event/osdi06/tech/full_papers/weil/weil.pdf

33. What is Amazon S3? (Amazon S3 data consistency model). Amazon S3 User Guide, Amazon Web Services, viewed 8 November 2026. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

34. PutObject. Amazon S3 API Reference, Amazon Web Services, viewed 8 November 2026. https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

35. Distributed Checkpoint: torch.distributed.checkpoint. PyTorch 2.15 documentation, PyTorch, last updated 8 July 2026, viewed 8 November 2026. https://docs.pytorch.org/docs/2.15/distributed.checkpoint.html

36. 2022 Ceph User Survey Results. Craig Chadwell, Ceph blog, ceph.io, 12 July 2022, viewed 8 November 2026. https://ceph.io/en/news/blog/2022/ceph-user-survey-results-2022/

About this report

Published by Rillor on 12 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Ceph block, object and file storage: which workload uses which. 12 November 2026. https://rillor.com/insights/ceph-block-object-and-file-storage

Notices

Rillor is not a registered investment adviser or commodity trading advisor and does not provide investment or trading advice. Research, forecasts, data and software described on this site are for research and engineering use. Nothing here is an offer or recommendation to buy or sell any security, commodity interest or digital asset. Past or simulated results do not indicate future results.

NVIDIA, AMD, Ceph, Kubernetes and related product names are trademarks of their owners and are used only to identify products. No affiliation or endorsement is implied.

Work with Rillor

Send what the storage must hold, how fast it grows, how it is read and written, and where it runs. Contact us at rillor.com/contact?type=storage or contact@rillor.com. More on distributed storage: rillor.com/compute/storage