## **CephFS Snapshots Evaluation**

Enrico Bocchi Abhishek Lekshmanan CERN IT - Storage

Ceph Dev+User Meeting 22 Feb 2024



# LHC

[Large Hadron Collider]

CMS

ALICE

LHCb

ATLAS







# **CERN Computing Infrastructure**

### **Embarrassingly Parallel, High Throughput Computing Lab:**

- Batch Farm based on HTCondor (270k+ cores)
- Physics Storage with EOS (~1 PiB on disks) + CTA (~700 PiB on tapes)

### IT Infrastructure Needs:



- Block Storage for OpenStack (1.7k hypervisors, 15k VMs) and DBs
- Object Storage for software distribution and cloud native applications
- File Storage for traditional networked FS and HPC scratch areas



# **Ceph at CERN**

### • IT Services:

- <u>Cloud Infrastructure</u>: OpenStack, K8s, OpenShift
- Code repositories, Container Registries, GitOps, Agile Infra
- Monitoring: Open Search, Kafka, Gafana, InfluxDB, Kibana
- Document Repositories // Web: Indico, Drupal, WordPress
- <u>Analytics:</u> HTCondor, Slurm, Jupyter Notebooks, Apache Spark

#### • Other Storage:

• NFS Filers, AFS, CVMFS, CERN Tape Archive, ...

### • Physics Experiments and End-Users:

- Accelerator Complex Monitoring
- Microelectronics Design
- Engineering and Beams





# **Ceph at CERN**

Application		Size (raw)	Version
RBD (OpenStack Cinder/Glance, krbd)	Production, HDDs	24.5 PiB	Pacific, Quincy
	643 TiB	Pacific	
Ну	265 TiB	Quincy	
CephFS (OpenStack Manila – K8s/OKD PVs, HPC)	Production, HDDs	12.6 PiB	Pacific, Quincy
	1.2 PiB	Pacific	
Hy	220 TiB	Quincy	
CERN Tape Archive (CTA)	Tape DB and Disk Buffer	235 TiB	Pacific
RGW, RGW + RBD Backup (2nd location)	Production (4+2 EC)	28.7 PiB	Pacific
RGW Multi-Site	Testing (4+2 EC)	4.2 PiB	Reef



# **CephFS** at **CERN**

#### • Use cases:

- General purpose networked FS for Open Infrastructure
- HPC scratch spaces: Highly parallel, fully-consistent POSIX FS for Beam simulations, Plasma Physics, Computational Fluid Dynamics, <u>ASICs</u>, ...
- Windows DFS replacement (exported over SMB)

### • Main consumers:

- Container based workloads K8s, OpenShift and the plethora of applications running on top...
- HPC scratch spaces and cluster-local users' homes
- Software building and distribution (Linux@CERN, Koji, GitLab)
- Monitoring, Web hosting, Documentation // Issue Tracking, Groupware
- Physics volunteer computing (BOINC) and Analysis Reproducibility Platforms





# **CephFS Operations**

#### • First production cluster started operation in 2013:

- Currently 4.2 PiB on HDDs, with metadata on SSDs 3.5k subvols, 3k+ clients, 10k+ reqs/s
- 1 FS, 4 active MDS (+ 4 stand-by, no standby-replay), no snapshots
- Explicit pinning of subdirs to a random MDS (+ a few selected users on dedicated MDS)

# do a consistent hash(\${dir}) modulo max\_mds
pin=`echo \${dir} | cksum | awk -v m=\${max\_mds} '{ print \$1 % m }'`
# pin the dir
echo Pinning \${dir} to \${pin}
setfattr -n ceph.dir.pin -v \${pin} \${dir}

- 2nd general purpose cluster added in 2020:
  - Entirely on flash storage (SATA SSDs, going to NVMe soon)
  - 1 FS, 1 active MDS (maybe going to multi-active in the future), no snapshots
- Other 4 CephFS clusters for various use cases:
  - 2x HPC scratch space and working directories (with standby-replay) for MPI clusters
  - 1x DFS replacement (CephFS kernel mount + SMB export, no vfs\_ceph)
  - 1x general purpose with snapshots



### **Purpose of this talk**

This is a report on our observations and measurements on the CephFS Snapshot functionality

- 1. Share our findings with other members of the Ceph community and learn from similar experiences
- 2. Discuss tools and strategies one can employ to mitigate issues
- 3. Address identified issue with guidance from upstream developers



# Why CephFS Snapshots?

- 1. Soft deletions (with self-service recovery from . snap dir)
- 2. Rollback to previous known state
- 3. Complete feature set of overlaying infra:
  - OpenStack Manila has snapshot support similar to RBD
  - Ceph-CSI for container orchestrators (K8s, OpenShift, ...)

#### Snapshots would immediately benefit existing use-cases:

- Residual migration out of NFS "Filers" (RBD + ZFS) to CephFS
- Versioning of software packages for gated releases, web hosting, HPC home dirs

#### ... and unlock new ones:

- Self-service DB provisioning (MySQL, PostgreSQL, InfluxDB)
- AI/ML workloads to checkpoint (and replay) stages for hyperparameter optimization



# Why CephFS Snapshots?

Increased effort on Business Continuity and Disaster Recovery
 See our full Cephalocon 2023 talk

Keep data safe facing physical infrastructure damage, human error, malicious attacks, ...

- Active-Active  $\Rightarrow$  Experimenting with Stretch Clusters
- Warm-Standby ⇒ CephFS Snapshots + Mirroring
- Backup and Restore ⇒ CephFS Snapshots guarantee crash-consistent backups and save MDS cycles w.r.t. live FS copy with external tool (e.g., rsync)

Apps on top can orchestrate to freeze IO and flush dirty cache pages to make snaps application consistent.



COST

**RECOVERY TIME** 

# Why Are We Hesitant?

- Non-decreasing number of strays, and assert during ~mdsdir scrub
  - Tracker <u>#51824</u>
  - Backported to Pacific (16.2.14), Quincy (17.2.7), Reef (18.2.1)
- PGs not snap trimming following scrub
  - Tracker <u>#52026</u>
  - Backported to Pacific (since 16.2.8), Quincy (17.2.7)
- MDS crashes with DB (PostgreSQL) flushes and snapshots, xlock state assert
  - Trackers <u>#57411</u> ⇒ <u>#44565</u>
  - Backported to Pacific (not tagged), Quincy (17.2.7), Reef (18.2.1)
- "High cephfs MDS latency and CPU load with snapshots and unlink operations"
  - Tracker <u>#53192</u>



# **Goal: Evaluate the Impact of Snapshots**

- CephFS usage at CERN dominated by (already-busy) general purpose clusters
- Heterogeneous workloads, both interactive and non-interactive
- If we were to provide snapshots as part of the service to our users...
  - Which workloads will suffer more?
  - Do they potentially affect cluster performance or functionality?
  - In case of degradation, will it be confined to a client? Subvolume? Entire FS?
- Understand operational impact and possible mitigations:
  - Focus on allowing users to snapshot their trees (and eventually backing these up)
    - Snapshots would be local to subvolumes needing them; No snaps for /
  - To a large extent, many users are not aware that they are using a shared FS (and we'd like to keep it that way :))



## **Experimental Setup**

### • Ceph cluster:

- Quincy 17.2.7
- 4 MDS (2 active, 2 standby)
- 3 OSD nodes
  - Each with 48 HDDs, 14 TB, 7200 rpm
  - Journals on 4x NVMe, 3.84 TB, PCIe 3.0
- Meta + Data replica size 3

### • Ceph clients:

- 4 clients mounting different paths
  - 2\* /volumes/\_nogroup/client
  - 2\* /volumes/\_nogroup/snapclient
- VMs with AlmaLinux 9
  - Fuse client at Quincy 17.2.7
  - Kernel 5.14.0-362.13.1.el9\_3
- snapclient workloads pinned to mds.1



## **Experimental Setup: Baseline stats**

- OSD Bench:
  - 4k objects: 2MB/s, 500 IOps
  - 4M objects: 250 MB/s, 60 IOps
- RADOS Bench:
  - BW: 930 MB/s ± 50
  - IOps: 240 ± 30

- IO500 Benchmark (16 worker procs):
  - BW: 1 GB/s (ior-easy-read/write)
  - MDTest:

	<u>Easy</u>	<u>Hard</u>
Write	939	750
Stat	1.44k	1.13k
Delete	650	320



# **Client Workloads**

- IO500:
  - Standard set of easy/hard benchmarks for IO performance measurements Write, Read, Delete
    - <u>IOR:</u> Generally creates large files with multiple workers writing to the same or different files
    - MDTest: Metadata-centric workload (file/directory tree creation), with or without data
  - Many tests are more metadata than BW intensive, which is good!
    - Expect potential bottlenecks on metadata with snapshots
  - Easy to add creating snapshots as a barrier between write and read/delete phases
- Linux Kernel untar/rm tests:
  - Continuous background untar+rm of Linux Kernel on both client setups (with and without snapshots)
    - On an average, untar takes ~3 minutes, rm takes ~4 minutes
  - Insight into how a repeated (and possibly in-cache) workload sees system performance over time



## **IO Benchmarks**

- **IO Write** benchmarks have similar performance both with and without snapshots:
  - Expected no impact on writes
  - Snapshots are taken after the write phase
- **IO Read** benchmarks show better performance on clients without snaps:
  - Unexpected
  - Maybe limited by what single procs can read?
- Similar observations can be made for:
  - IOR Hard tests
  - MDTest {easy, hard} {read, stat}
- MDTest delete performs better without snapshots





## **Stress Benchmarks with MDTest**

• Let's stress the system with more metadata intensive workloads!

- On the clients without snapshots, we created a MDTest benchmark with a deep directory tree
  - Degradation of removal times proportional to the tree depth
  - Overall system parameters still good
- Repeated tests on client with snapshots, with a snapshot barrier between create and delete phases,

lead to a cluster-wide impact





## **Stress Benchmarks with MDTest**

- The MDS serving subvol with snapshots showed increasing latency and eventually failed to send heartbeats:
  - A standby MDS takes shortly after, but stays in replay for a few tens of minutes
  - The entire cluster becomes unusable in this phase
  - Once recovered, all client traffic seemingly reaches mds.0 only, regardless of pins







# **Stress Benchmarks with MDTest**

- MDS kept busy by CInode::is\_ancestor\_of
- Any other IO activity from non-snap client also suffers badly
- Workloads started by clients never complete
- Worst case IOR/MDTest benchmark yielded 25 MB/s < 25 IOps; Hard tests never completed
- Worst case untar+rm took over 4h+ with the cluster not really doing any IO...
- Observations consistent
   with <u>Bug #53192</u>

Samples: 61K	of event	'cpu-clock:ppp	H', Event count (approx.):	61258258197				
Children	Self	C <mark>ommand</mark>	Shared Object	Symbol				
- 94.39%	0.00%	MR_Finisher	[unknown]	[.] 000000000000000				
- 0		_•						
• 84.6	• 84.65% CInode::is_ancestor_of							
9.43	% SnapRea	lm::split_at	. 2					
+ 84.65%		MR_Finisher	ceph-mds	[.] CInode::is_ancestor_of				
+ 9.43%		MR_Finisher	ceph-mds	[.] SnapRealm::split_at				
+ 0.72%	0.00%	log	[kernel.kallsyms]	[k] entry_SYSCALL_64_after_hwframe				
+ 0.72%	0.05%	log	[kernel.kallsyms]	[k] do_syscall_64				
+ 0.63%	0.04%	log	libpthread–2.28.so	<pre>[.] pthread_cond_wait@@GLIBC_2.3.2</pre>				
0.51%	0.02%	log	[kernel.kallsyms]	[k]x64_sys_futex				
0.50%	0.01%	log	[kernel.kallsyms]	[k] do_futex				
0.46%	0.02%	log	[kernel.kallsyms]	[k] futex_wait				
0.43%	0.00%	ms_dispatch	[unknown]	[.] 00000000000000				
0.42%	0.00%	log	[kernel.kallsyms]	[k] futex_wait_queue_me				
0.41%	0.00%	log	[kernel.kallsyms]	[k] schedule				
0.41%	0.00%	log	[kernel.kallsyms]	[k]sched_text_start				
0.41%	0.00%	log	[kernel.kallsyms]	[k] finish_task_switch				
0.41%	0.41%	log	[kernel.kallsyms]	[k]raw_spin_unlock_irq				
0.39%	0.06%	ms_dispatch	libpthread–2.28.so	[.] pthread_cond_broadcast@@GLIBC_2.3.2				
0.37%	0.00%	ms_dispatch	[kernel.kallsyms]	[k] entry_SYSCALL_64_after_hwframe				
0.37%		ms_dispatch	[kernel.kallsyms]	[k] do_syscall_64				

# Linux untar/rm: Before and After

- Reminder: Linux Kernel untar+rm kept going in background while running stress tests
- The stress workload was applied twice a day for a short period of time
  - The MDTest itself only took a few minutes
- Clear trend in untar+rm taking longer:
  - Tangible shift in the total time to execute
  - Outliers are expected; Some exceeded 1h (and were excluded from the graph)



 ${\sim}500$  samples per op before & after



## **Other issues observed**

• After the stress workloads were removed,

the MDTest benchmark from the snap client is in a permanently stuck state

- Many PGs in snaptrim (or snaptrim\_wait) state:
  - Ever-increasing number of Log Segments on mds.1, eventually going OOM
  - Unclear how to help the MDS trimming faster
  - Tried tuning the following parameters to no avail
    - mds\_log\_max\_segments
    - osd\_recovery\_sleep\_hdd
    - osd\_snap\_trim\_sleep\_hdd
    - osd\_pg\_max\_concurrent\_trims
  - MDS in replay state for a week now:
    - Process spinning 100%, but very little IO
    - Added 100GB swap file to avoid OOMs





# **More Takeaways from the Stress Tests**

• Non-trivial effect on non snapshot users is a general concern:

- We saw cluster workloads (both interactive and non-interactive) being severely affected when deletion workloads are triggered
- Not easy (yet) to monitor the potential triggers for MDS
- Further investigation on why single MDS seem to take all IO traffic after switch:
  - During all the MDS switchovers triggered by snap deletes, we observed standby MDS switching over
  - All the MDS snap workloads were pinned to MDS.1, however we saw general operations not statically pinned all land on one MDS
  - Consistent pattern where one active MDS is completely busy and the other completely free
- Pinning away snapshots workloads didn't seem to help much:
  - While wasn't intended to solve the problem for the snap workloads, we expected non-snap workload operations not metadata intensive to generally work closer to baseline



### Conclusions

- We are *very happy* with Ceph:
  - Blocks, Objects, Files from a single platform
  - Similar hardware, cluster administration/maintenance, tooling, ...
- We are also *very happy* with CephFS, and so are our users!
- Snapshots are a nice addition (likely a must-have from BC/DR requirements), but still scary for large-scale production cluster:
  - We do not want a fraction of users impacting the whole service
  - Some observed behaviors not fully understood (all reqs going to one MDS, eternal replay, performance shift)
  - We'd love to hear from you and your experience with snapshots in production ...or if you've seen any of the issues we've seen
  - We'd love to contribute with community and upstream developers to address identified issues



