CephFS: a new generation storage platform for Australian high energy physics

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 062015 doi:10.1088/1742-6596/898/6/062015

CephFS: a new generation storage platform for Australian high energy physics

G Borges[1], S Crosby[2] and L Boland[2]

[1] ARC Centre of Excellence for Particle Physics at the Terascale, School of Physics A28, The University of Sydney 2006, New South Wales, Australia
[2] ARC Centre of Excellence for Particle Physics at the Terascale, School of Physics, level 3, 258 Queensberry Street, The University of Melbourne 3010, Victoria, Australia

E-mail: goncalo.borges@sydney.edu.au

Abstract. This paper presents an implementation of a Ceph file system (CephFS) use case at the ARC Center of Excellence for Particle Physics at the Terascale (CoEPP). CoEPP's CephFS provides a posix-like file system on top of a Ceph RADOS object store, deployed on commodity hardware and without single points of failure. By delivering a unique file system namespace at different CoEPP centres spread across Australia, local HEP researchers can store, process and share data independently of their geographical locations. CephFS is also used as the back-end file system for a WLCG ATLAS user area at the Australian Tier-2. Dedicated SRM and XROOTD services, deployed on top of CoEPP's CephFS, integrates it in ATLAS data distributed operations. This setup, while allowing Australian HEP researchers to trigger data movement via ATLAS grid tools, also enables local posix-like read access providing greater control to scientists of their data flows. In this article we will present details on CoEPP's Ceph/CephFS implementation and report performance I/O metrics collected during the testing/tuning phase of the system.

1. Introduction

The ARC Centre of Excellence for Particle Physics at the Terascale (CoEPP) was established in 2011 as a collaborative research venture between the University of Melbourne, the University of Adelaide, the University of Sydney and Monash University. It provides a common umbrella for the coordination of High Energy Physics activities, bringing together theoretical and experimental physicists working within Australia. CoEPP also enables the Australian participation in the ATLAS experiment at the Large Hadron Collider (LHC) at CERN in Geneva, and in the BELLE II experiment at SuperKEKB in Japan.

The distributed nature of CoEPP's centres raises a difficult challenge on how to provide computing and storage services satisfying everyone, everywhere. Each CoEPP centre was initially equipped with a dedicated NFS server (∼60TB) and a small pool of local physical computing resources for interactive workloads and batch system submissions. Researchers soon realized that the computing capacity available to them was not sufficient for processing their workloads in a timely manner. This issue was addressed through collaboration with the National eResearch Collaboration Tools and Resources project (NECTAR) [1], an Australian governmental funded body which operates an OpenStack computing service. NECTAR granted access to a maximum of 1400 virtual processors for CoEPP activities. Virtual machines were

built and configured as CoEPP resources and integrated in a dedicated batch system. The system was later enhanced with an elastic central computing service (DynamicCluster) [2] capable to scale up or down virtual machines according to the workload demand. This cloud-based service rapidly became CoEPP's primary computing infrastructure.

The fact that researchers could now access a larger set of computing resources emphasized a second problem: how to share and process data stored elsewhere? Soon it became evident that copying large data sets (ranging from hundreds of GBs to some TBs) between CoEPP centres and the cloud service was not an effective solution. This highlighted the need to provide a shared storage infrastructure accessible to anyone, everywhere. The problem was the lack of funding to support the acquisition of dedicated resources for this purpose. An opportunity came along with the acquisition of larger storage capacity servers and the release of out-of warranty resources under the yearly review of the Australian Tier-2 capacity for ATLAS. The challenge was now on how build a performant distributed storage system with commodity hardware.

2. State of the Art: distributed storage systems

The focus of our work is on products that address, first and foremost, individual researchers needs. For this reason, we excluded products not designed for a traditional use, as common HEP grid solutions (DPM, dCache and others) or Hadoop's file system (HDFS) which, in general, do not offer the best user experience nor can be directly mounted for a user to view.

IBM GPFS / Spectrum scale [3] is a distributed parallel file systems widely used in high performance computing with exceptional performances. The need to keep costs under control eliminated this solution due to its expensive license.

Lustre filesystem [4] is a mature free open source product normally offered as an alternative to GPFS. However, we questioned its capability to operate with commodity hardware. Lustre high availability operation relies on host-based failover: two metadata servers (MDS) configured as an active/passive pair, and object storage servers (OSS) deployed in an active/active mode (to provides redundancy without extra overhead). This solution will not scale in an environment where we expect a high number of hard drives failures (rather than host failures).

Red Hat offers two different open source solutions: Gluster [5] and Ceph [6]. Gluster was primary developed as a scale-out file storage solution that adds extensions for object storage. Ceph is a highly scalable object storage product with block and file capabilities. Both use a hashing algorithm, exported to all servers, to place data that allows them to figure out where a particular data item should be kept. As a result, both systems can easily replicate data. Moreover, the lack of central metadata files means that there is no bottleneck in file access. It is often argued that Gluster should be better as a scale-out NAS and that Ceph would better fit object storage applications and block storage provisioning (for example, in the cloud). Nevertheless, both solutions seem to cover the same kind of use cases. The major technical difference between the two is how they handle replication and rebalance after failures. Ceph is able to identify problematic storage devices, mark them down and automatically replicate the data held there (using copies from other healthy devices). This automatic self healing process is managed by a highly customazible and configurable algorithm. In Gluster, the policy replica placement into failure domains must be manually defined by the administrator.

Overall, Ceph seemed more appropriate to work with commodity hardware due to its self recovering and healing behaviour. Moreover, HEP is represented in Ceph community bodies (through CERN) providing a direct link to raise requirements and guarantee they are addressed in a timely manner. Given all these considerations, we opted to further investigate Ceph.

2.1. Ceph overview

Ceph is a cutting edge, open source, distributed data storage technology. It is based on self-healing, intelligent object storage devices (OSDs), a combination of CPU, network interfaces,

local cache and underlying disk space. An even set of lightweight monitors are responsible for monitoring the state of each OSD and ensure a highly available and healthy distributed object store cluster (called RADOS). Applications can directly store, retrieve or delete objects from RADOS through LIBRADOS, a library offered in many different implementation (C, C++, Java, Python, Ruby and PHP). LIBRADOS stripes the data by writing byte ranges to predictable named objects, grouped in placement groups, and delivered to OSDs according to a specific but configurable algorithm (named CRUSH). Ceph also provides standard clients, which by embedding LIBRADOS functionality, allow an easier and tranparent access to the object store:

- RADOS Gateway (RGW): an HTTP REST gateway for RADOS object store.
- RADOS Block Devices (RBD): a reliable fully distributed block-based storage interface. RBD is supported in Qemu, and heavily used for the provision of virtual machines in cloud-based systems like OpenStack and CloudStack (that rely on libvirt and Qemu).
- CephFS: a distributed posix-like file system with scale out metadata management. At the time of writing, only one active metadata server can be deployed. However, CephFS promises to deliver a cluster of metadata servers dynamically partitioning responsibility for the file system namespace and distributing the metadata workload based on client accesses.

Ceph is gaining popularity across the Worldwide LHC community with many OpenStack, RGW and RBD deployments and a growing usage for physics activities. At CERN Tier-0, a RGW is used to enable a RADOS storage back-end for OpenStack Cinder and Glance services. Other use cases are the provision of NFS (∼50 TB across 28 servers) or CVMFS stratum 0 services under RBD. On the grid front, there has been community work focused on developing RADOS plugins for GRIDFTP services and a bespoke HEP protocol called XROOTD.

CephFS has not been subject to the same level of interest partially because it is a less mature product. However, it delivers a set of functionality which may have a big impact on physics analysis, especially for local research activities:

(i) CephFS represents a drop in replacement for any kind of local or network file system.

(ii) Data and metadata are directly stored in RADOS, and therefore, can easily scale up with the object store size. This is a considerable enhancement with respect to traditional posix file systems, specially for the metadata, where the maximum number of inodes is constrained by the size of the metadata partition, and can be quickly exhausted.

(iii) Different stripping layouts can be defined at the directory level using extended attributes. This provides a flexible mechanism to fine tune how the data should be stored under different (application) use cases. Clients can write a (customizable) number of stripe units of data (with a customizable size) to their corresponding objects in parallel. Since objects get mapped to different placement groups and further mapped to different OSDs, each write occurs in parallel at the maximum write speed.

(iv) Ceph provides two types of CephFS clients: a kernel module and a fuse client. The kernel module provides higher performance. However, the fuse client is a flexible way to access the file system without having to rely on functionality only available in modern kernels. This is of extreme value on the slow evolving HEP systems where the transition to more modern environments is heavily constrained by experimental restrictions.

3. CoEPP's CephFS: On the road to production

Our testing infrastructure consisted of a small pre-production Centos7 RADOS cluster, version 9.2.0 (Infernalis), with 3 monitors and 32 x 3 TB RAID-0 SAS disks (8 per storage server) deployed as OSDs with data and journal partitions collocated in the same devices. Ceph storage nodes were connected to a 10 Gb/s NIC, both on a public network as well as on a private storage network. The CephFS layer was added through the deployment of a metadata server with 32 GB

of memory. The data and metadata pools were created with a 3x replication level, and the file system mounted on a 10 Gb/s Scientific Linux 6 server using a long term support kernel.

Because researchers main interest was in Ceph file system capability, our testing activities focused on CephFS. Since our hardware setup was not optimized (for example, OSD journals were not deployed on SSDs as suggested by best practices), the underlying idea was to study CephFS performance as a function of its stripping layout parameterizations more than to get an upper limit for I/O bandwidth. We used the Flexible IO tester (aka fio) [7] to probe CephFS performance. The client cache was purged before each test and all fio runs were executed with ioengine=libaio, iodepth=64 and direct=1. Fio tests consisted of writing or reading 8 GB files of data, with variable fio block size (from 4 KB to 512 KB), to and from CephFS directories with different layouts. Three customizable parameters define the directory layout: object size, stripe size and stripe count. The object size was kept fixed to 4 MB (its default value). This choice was supported by preliminary RADOS benchmark tests. The stripe unit and stripe count varied from 64 KB to 512 KB, and from 2 to 8, respectively.

Figure 1 presents the measured bandwidth for single threaded fio write runs, using different fio block sizes, as a function of CephFS's directory stripe unit. The performance for sequential and random write operations are shown in the figure's left and right plots, respectively. Figure 2 presents the equivalent measurement for read operations. We observe a correlation between the stripe unit and the application block size, i.e., a growth of the bandwidth as the stripe unit approaches the fio block size, followed by a saturation at its maximum value for stripe unit $\geq$ fio block size.

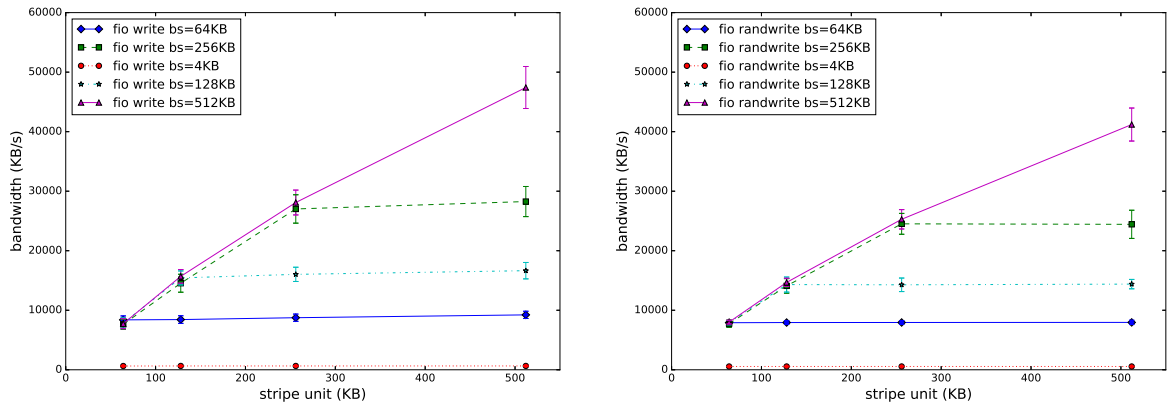

Figure 1. Bandwidth as a function of CephFS stripe unit (with fixed object size $=\dot{4}$ MB and stripe count $= 8$) for single threaded sequential (left) and random (right) fio write operations.

Figure 3 presents the measured bandwidth for single threaded fio runs as a function of CephFS's directory stripe count. The left plot superimposes the results for sequential and random write operations while the right plot superimposes the results for sequential and random read operations. All measurements were executed using a fixed value (512 KB) for the fio block size and the stripe unit which according to previous results, maximizes performance. We do not observe a dependency of the bandwidth values as a function of the stripe count. This observation contradicts the current literature where one might expect a significant performance by increasing the stripe count. The fundamental argument is that clients will write the stripe units to their objects in parallel. By spreading that write over multiple objects (which map to different placement groups and OSDs) Ceph can reduce the number of seeks per drive and combine the throughput of multiple drives to achieve faster write (or read) speeds. Finally, we observe a slight performance degradation when comparing random with sequential operations: $\sim$13% and $\sim$4% for write and read operations.

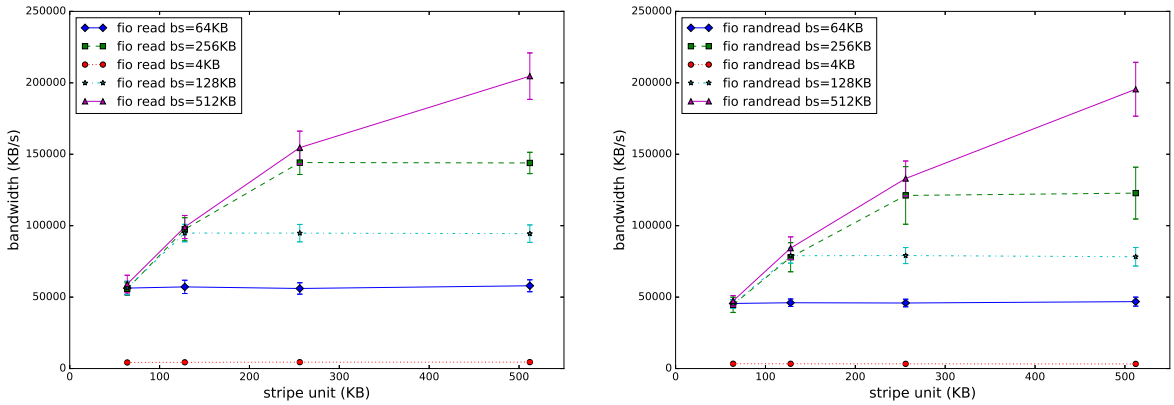

Figure 2. Bandwidth as a function of CephFS stripe unit (with fixed object size = 4 MB and stripe count = 8) for single threaded sequential (left) and random (right) fio read operations.

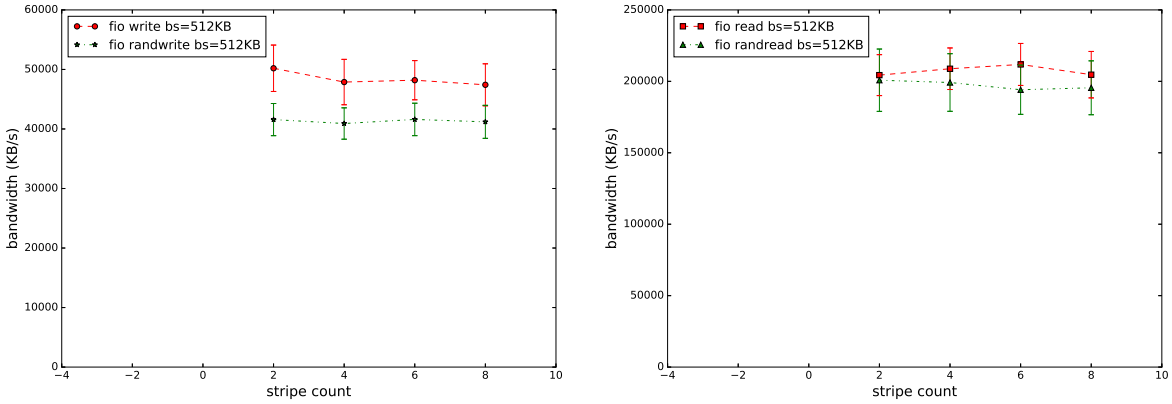

Figure 3. Bandwidth as a function of CephFS stripe count (with fixed object size = 4 MB and stripe unit = 512 KB) for single threaded fio runs. The results for sequential and random oepration are superimposed for writes (left) and read (right) fio tests.

Figure 4 presents the measured aggregated bandwidth for multi-threaded fio runs as a function of the number of threads, in sequential and random mode, for write (left) and read (right) operations. These results show that the aggregated bandwidth scales as a function of the number of threads. Up to 128 fio threads, the maximum value tested, the aggregated bandwidth does not give signs to saturate. In the read cases, it seems near the theoretical value of 1.25 GB/s.

To assess the robustness of the system, we stressed our pre-production infrastructure with a wide set of crash tests: simulation of OSD service failures, OSD file system problems, OSD journal failures, storage server failures and connectivity issues. Ceph was always able to recognize problems and automatically trigger data recovery from other healthy instances. However, it was also clear that the detection of the failure was not immediate, sometimes happening a few hours later. Moreover, without knowing *à priori* the source of the problem, its diagnosis is time consuming due to the distributed nature of the system, the high amount of generated logs, their complexity, and the necessity to correlate events in an effective way.

With the focus to deliver higher storage capacity with greater performance, we upgraded our system to Ceph 10.2.2 (Jewel, the latest major release) and expanded our infrastructure to

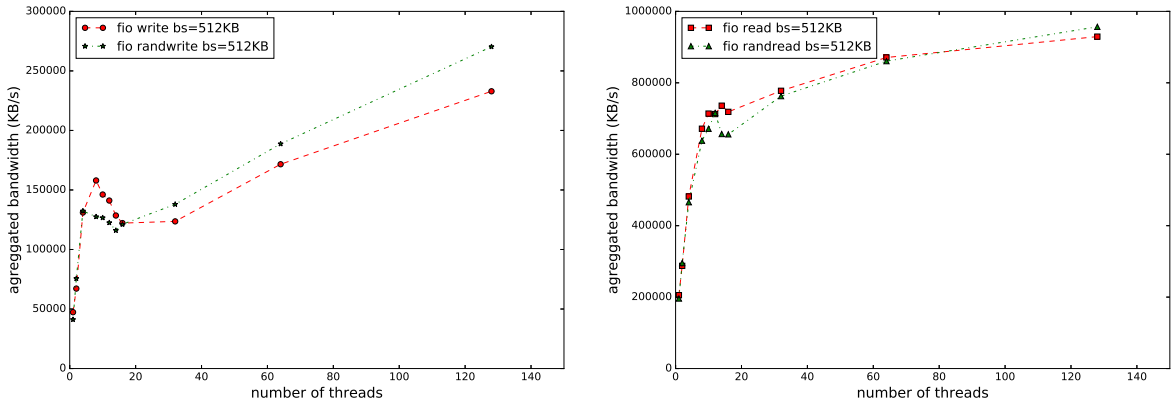

Figure 4. Aggregated bandwidth as measured by a variable number of fio threads during sequential and random write (left) and read (right) operations. The CephFS's directory layout was set with an object size = 4 MB, stripe unit = 512 KB) and stripe count = 8.

11 storage servers providing 112 x 3 TB OSDs (with 4 GB of RAM per OSD). A fundamental difference with respect to the pre-production setup was the decoupling of OSD journals from data partitions in the same device. OSDs journals were deployed in Solid State Disks (4 journal partitions per SSD). For a higher CephFS availability, we deployed a standby-replay MDS.

During the pre-production phase, it became clear that the CephFS kernel client was not a flexible solution. Users' experimental collaborations only provide analysis software bundles certified for Scientific Linux 6 (SL6). On the other hand, mounting CephFS using a kernel module requires kernel version 2.6.34 or later, but SL6 distributions only release version 2.6.32. An alternative was the installation of a long term kernel (as done for our pre-production testing). However, kernel clients are always behind in terms of bug fixes and new functionality because it takes time to push new developments through the kernel release cycle. For these reasons, we use ceph-fuse at the cost of slightly lower performance. A recent study from ebay [8] demonstrated that ceph fuse was 3-4 times slower than the kernel client while completing 10000 *"touch"* operations of new files, and 7-8 times slower while creating 4096 (1KB) files (using O_DSYNC). Although less performant, it is more flexible (works in user space), reliable (synced with latest developments) and portable (easily patched, recompiled and deployed). Starting from the Infernalis major release, Red Hat dropped the support for RHEL 6 derivatives, including Scientific Linux 6. Without main stream binaries, we were forced to build ceph-fuse and all its (non-SL6) dependencies. This bundle was installed in CoEPP's CVMFS stratum 0, served to hosts via CVMFS clients, and activated via environment modules using puppet.

CoEPP's Ceph / CephFS core infrastructure is based in Melbourne, and more than 150 ceph-fuse clients are distributed among CoEPP centres and NECTAR zones spanning the Adelaide, Melbourne and Sydney regions. AARNET (the Australian research network provider) connects state capitals with a Nx100G dense wavelength division multiplexing (DWDM) network, and the major Australian universities are served by 40 GB points of presence (PoP), guaranteeing high standard connectivity. To assess how researchers experience CephFS over the WAN, we isolated a 4-core (16 GB RAM) virtual machine running in Nectar Sydney zone (the longest network path) with an average network latency of 13-14 ms (as reported by ping). Several bi-directional iperf (60 s) measurements showed bandwidth values fluctuating around 1 Gb/s, with maximum and minimum values of 1.6 Gb/s and 0.7 Gb/s, respectively. It is important to note that we do not control the virtual machine hypervisor meaning that other virtual machines hosted in the same hypervisor may interfere with the measurements. Table 1 presents aggregated

bandwidth and latency measurements, obtained running fio, for an NFS system (the legacy CoEPP solution) and CoEPP's new CephFS system. All fio runs were executed with bs=4M, ioengine=libaio, iodepth=64 and direct=1. Cache effects were minimized using 32 threads and 2 GB files per thread (4 times the available RAM). We conclude that while write operations are equivalent between both systems, Ceph provides a slightly better bandwidth and no performance degradation between sequential and random read operations, at the expense of higher latency.

Table 1. NFS and CephFS average aggregated bandwidth (MB/s) and thread latency (ms) for a 32 thread fio test running from a virtual machine in Nectar Sydney zone.

	write (lat)	randwrite (lat)	read (lat)	randread (lat)
NFS [MB/s]	127 (65.4 ms)	129 (64.3 ms)	191 (11.9 ms)	122 (18.5 ms)
CephFS [MB/s]	131 (52.1 ms)	112 (60.5 ms)	205 (34.8 ms)	205 (35.1 ms)

3.1. ATLAS Tier-2 integration

CoEPP's CephFS is also used as the back-end file system of an ATLAS user area (a second localgroupdisk) at the Australian Tier-2. Its integration into ATLAS FAX, a federated storage systems based on XRootD, is enabled by an XRootD server ability to access a posix file system. Client software tools like ROOT or xrdcp can be used to read data from this storage service regardless of location. On the other hand, the support for ATLAS Distributed Data Management (DDM) operations is provided through the deployment of StoRM, a Storage Resource Manager that relies on a posix file system back-end. StoRM will manage all data transfers from and to this space. Moreover, it allows us to define default ACLs, a list of ACL entries that will be applied automatically on each read (srmPrepareToGet) and write (srmPrepareToPut) operation (see figure 5). Through this functionality, it is possible to allow local access to files or directories to groups different from the one that made the SRM request. A user can then trigger data to be copied via Rucio (Web UI or CLI), monitor the transfer by checking the presence of files as well as their sizes, and later analyse these data by submitting batch jobs to their local clusters.

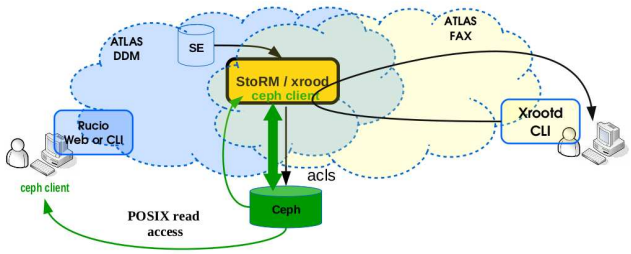

Figure 5. CoEPP's CephFS architecture for ATLAS DDM and ATLAS FAX interoperability.

4. A year of CoEPP's CephFS in production

CoEPP's CephFS operation has not been fully transparent. The following issues were identified:

- When multiple clients read and write files simultaneously, all caches must be disabled and all operations should go directly to the OSDs, to avoid clients from reading staled data. Ceph-fuse can dinamically disable its own cache but the kernel pagecache can only be turned off by setting 'fuse disable pagecache = true' in the configuration file.
- Ceph-fuse allocates more virtual memory than it is actually used. Our ceph-fuse client was compiled against glibc (>1.10) malloc function, which introduces memory pools called arenas (8 per core in 64 bits machines). Each memory pool takes 64 MB of address space. As

a result, our 24 core hosts were showing 8 x 24 x 64 MB = 12288 MB of used virtual memory. We decreased the number of arenas (from 8 to 4) with no visible performance impact by setting MALLOC_ARENA_MAX environment variable before starting ceph-fuse.

- Ceph-fuse clients are sensitive to random network perturbations and can enter a stale state. For the client to tolerate better short term connectivity issues, fine tune metadata server settings, such as 'mds session timeout', 'mds reconnect timeout' and 'mds session autoclose'.
- Ceph-fuse is made available to hosts via cvmfs. When a machine is reaching its memory limits, cvmfs processes are eligible to be killed by Linux OOM breaking the ceph-fuse client.
- Ceph Jewel 10.2.2 (our current version) does not properly handle listings of directories with high number (millions) of files leading to craches of the metadata server. Ceph 10.2.3 partially mitigates this issue by introducing a limit on the number of files per directory (100000). Future releases will solve this problem through directory fragmentation.
- Clients are frequently 'failing to respond to cache pressure'. This is a message displayed by Ceph to alert that one or more clients are not releasing inodes from their caches as requested by the metadata server. Our investigations showed non-used clients unable to release their cached data. This points to a bug in the ceph-fuse client.
- The metadata server uses memory pools for memory allocation, and it doesn't (currently) release memory back to the OS because it's doing its own cache size enforcement. When the cache size limits aren't being enforced this becomes a problem.
- It is not uncommon to only see Ceph(FS) bugs fixed on the eve of major releases. This is a problem for production systems where the main wish is to concentrate in addressing critical problems without introducing new functionality.

5. Summary and Conclusions

CoEPP's CephFS has been in production for more than a year. It is the chosen solution to address researchers' requirements toward more transparent data access and availability between CoEPP centres. In a pre-production environment, it was possible to understand how to fine tune CephFS directory layouts for particular use cases. Our bandwidth measurements showed a strong correlation between directory stripe unit and the application block size, reaching maximum throughput when the application block size matches the stripe unit. The product also showed itself as robust being able to automatically deal with disk, host and connectivity failures. CoEPP's production Ceph / CephFS core infrastructure is based in Melbourne, and more than 150 ceph-fuse clients are distributed among centres spanning the Adelaide, Melbourne and Sydney regions. The system performance is able to match (and even supersede) the previous NFS legacy solution. The cost is a slight higher I/O latency experienced by users due to the ceph-fuse client use in WAN mode, not critical under the current use case, where the data is used as input and output of long runtime batch processing jobs. After its first six months in operation, we are continuously increasing the system capacity due to researchers heavy use.

References

[1] National eResearch Collaboration Tools and Resources project (NECTAR) https://nectar.org.au/
[2] Zhang S. 2014 *J. Phys.: Conf. Series* **513** doi:10.1088/1742-6596/513/3/032107
[3] IBM white paper *An introduction to IBM Spectrum Scale*
[4] Lustre filesystem http://lustre.org/
[5] GlusterFS https://www.gluster.org/
[6] Weil, Sage A. 2006 Ceph: A scalable, high-performance distributed file system *Proc. 7th Symp. on Operating systems design and implementation* Seattle, Washington, USA.
[7] Flexible IO tester, git://git.kernel.dk/fio.git
[8] Cephfs jewel mds performance benchmark, https://www.slideshare.net/XiaoxiChen3/cephfs-jewel-mds-performance-benchmark