CoEPP
ARC Centre of Excellence for Particle Physics at the Terascale

CephFS: a new generation storage platform for Australian High Energy Physics

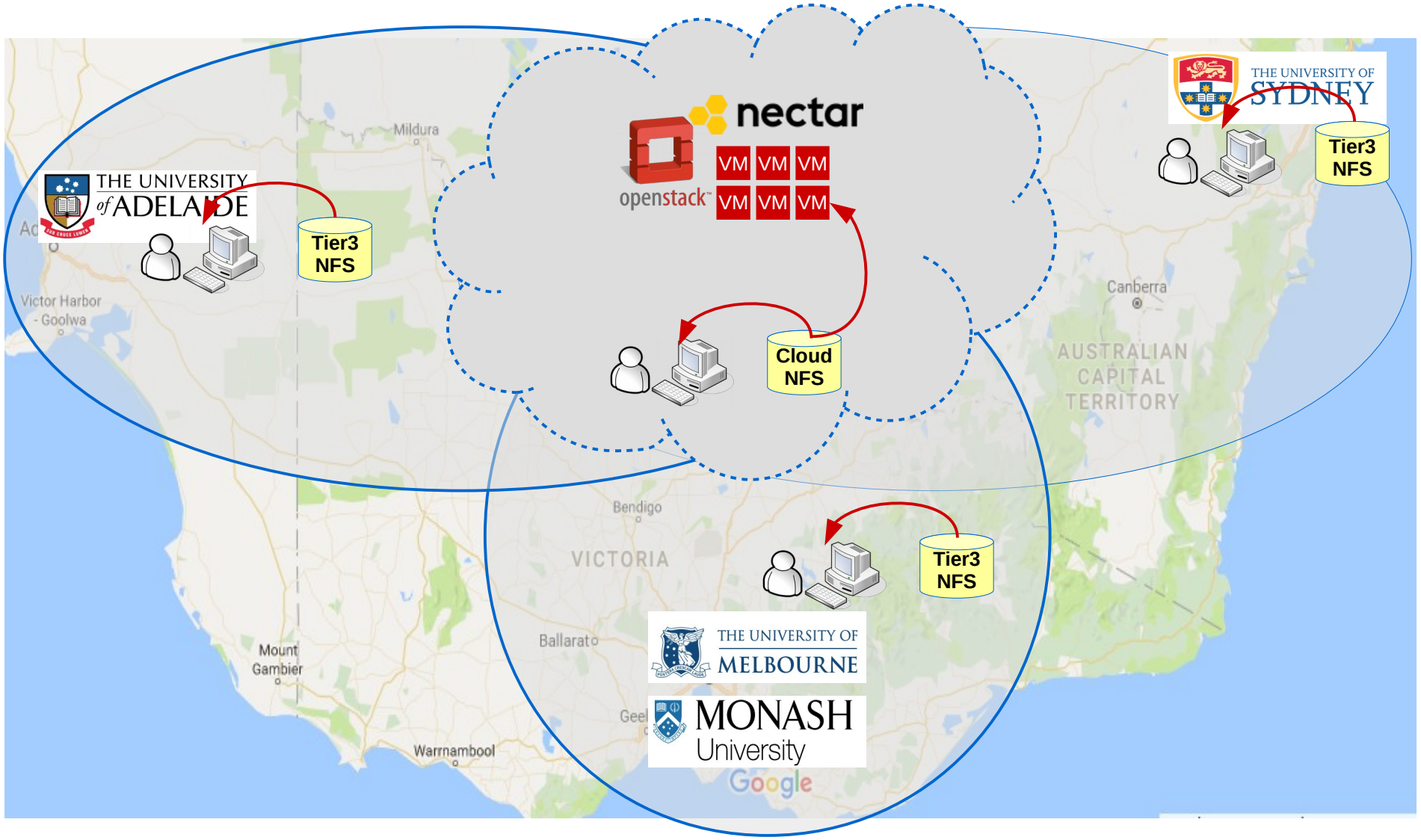
(an infrastructure / technical view)

Goncalo Borges (goncalo.borges@coepp.org.au)

Sean Crosby, Lucien Boland, Jeremy Hack

CoEPP
ARC Centre of Excellence for Particle Physics at the Terascale

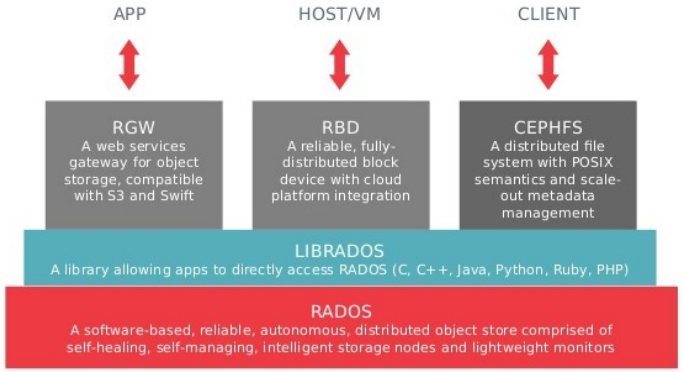
'Tier-3' services



HEPiX

Why Ceph / CephFS ?

- **Why Ceph?**
  - Open source / Community driven
    - HEP involvement already on-going
  - Designed to work with commodity hardware
    - Self recovering / self healing behaviour
    - No single point of failure
  - High granularity for customizing data operations
  - Different data access methods
  - Integration with other infrastructures (libvirt, openstack, ….)



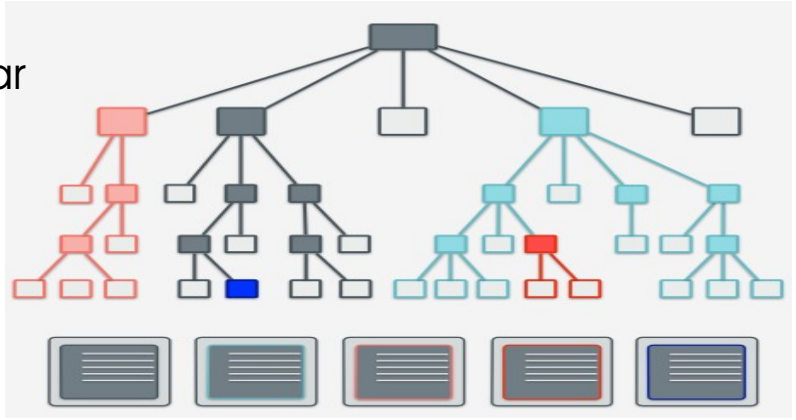
- **Why CephFS ?**
  - The POSIX-like requirement
  - The capability to make the filesystem available in different geographical locations
  - The metadata in RADOS

- **Jewel 10.2.X (X = 0,1,2,3)**
  - CephFS **stable** release (note that stable ≠ production)

    http://docs.ceph.com/docs/master/cephfs/best-practices/

CephFS features

- **Cooperative Partitioning between MDS servers**
  - Keep track of how hot metadata is
  - Migrate subtrees to keep heat distribution similar
  - Maintains locality

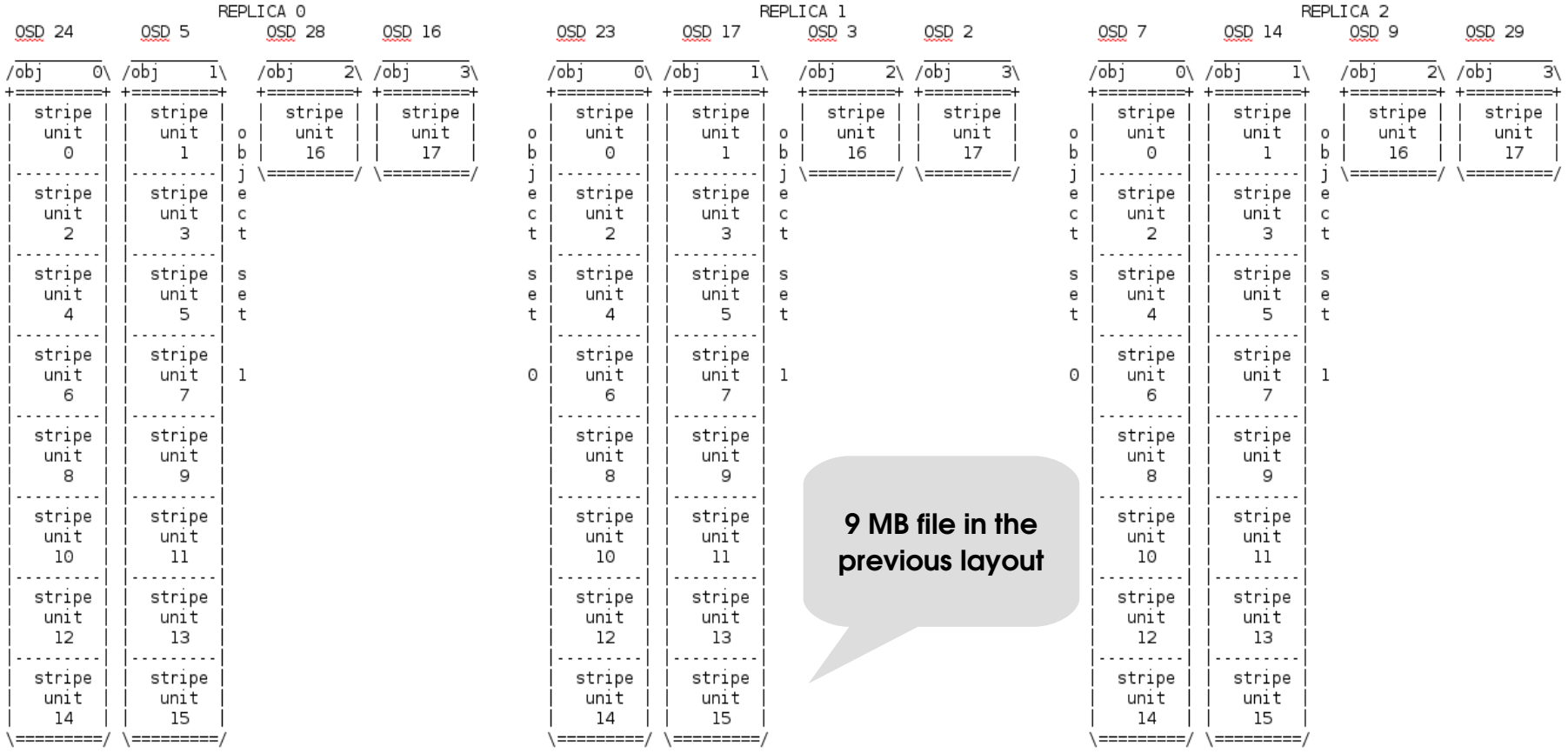


- **CephFS layouts and file striping**
  - Possible to define different layouts, using different pools, for different use cases
  - Layouts are defined as extended attributes at a directory level
  - Client writes the stripe units to their corresponding objects in parallel
  - Since objects get mapped to different placement groups and further mapped to different OSDs, each write occurs in parallel at the maximum write speed

```
# getfattr -n ceph.dir.layout /cephfs/objectsize4M_stripeunit512K_stripecount2
ceph.dir.layout="stripe_unit=524288 stripe_count=2 object_size=4194304 pool=cephfs_dt"
```

CephFS features

```
# getfattr -n ceph.dir.layout /cephfs/objectsize4M_stripeunit512K_stripecount2
ceph.dir.layout="stripe_unit=524288 stripe_count=2 object_size=4194304 pool=cephfs_dt"
```



CephFS features

- **Recursive statistics via extended attributes**

```
# cd /coepp/cephfs/

# ll
total 4
drwxr-xr-x 1 root root  1758765193050 Oct 13 05:06 adl
drwxr-xr-x 1 root root  1800679778770 Feb 16  2016 borg
drwxr-xr-x 1 root root 44364755984613 Jul 19 23:31 mel
drwxr-xr-x 1 root root              0 Feb 29  2016 mon
drwxr-xr-x 1 root root     8779587926 Jul 12 05:46 share
drwxr-xr-x 1 root root  2136590185337 Feb 10  2016 storm
drwxr-xr-x 1 root root  1388135221023 Jul 21 06:27 syd

# getfattr -d -m ceph share
# file: share
ceph.dir.entries="3"
ceph.dir.files="0"
ceph.dir.rbytes="8779587926"
ceph.dir.rctime="1468410021.09487269127"
ceph.dir.rentries="284"
ceph.dir.rfiles="189"
ceph.dir.rsubdirs="95"
ceph.dir.subdirs="3"
```

CephFS pre-production tests

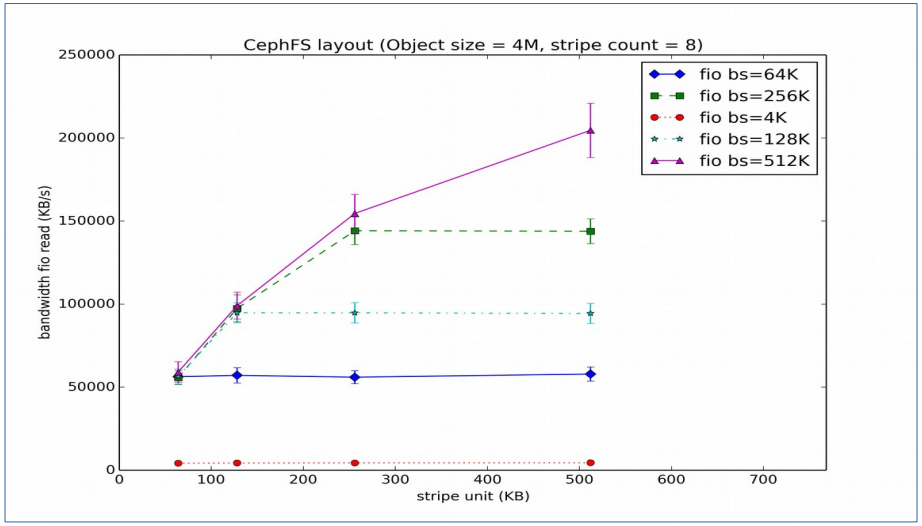
- **'Pre-production setup'**
  - 4 Dell R620 x 8 (3TB) OSDs (9.2.0)
  - Jornals in a separate OSD partition
  - A single host mount cephfs (kernel)
  - Single MDS server (32 GB RAM)

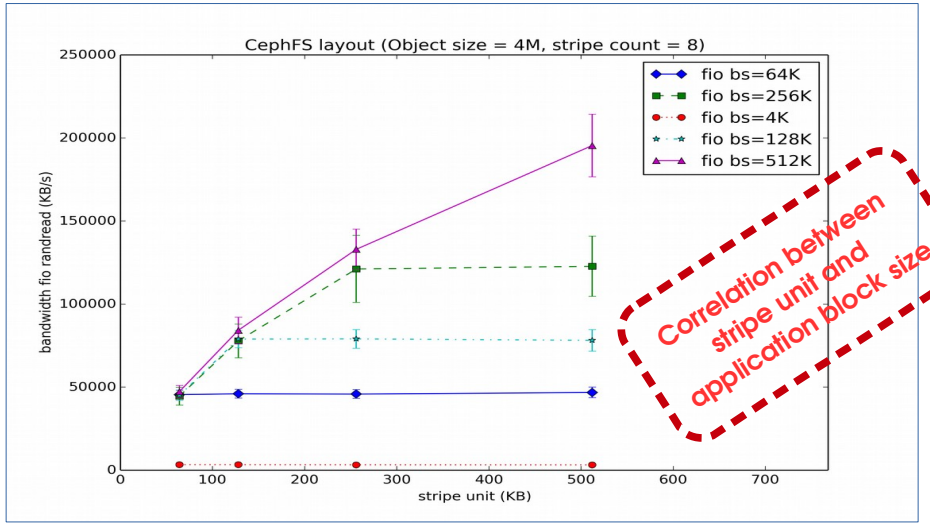
- **RADOS benchmarks**
  - Ceph is very good to store big objects and bad for small objects

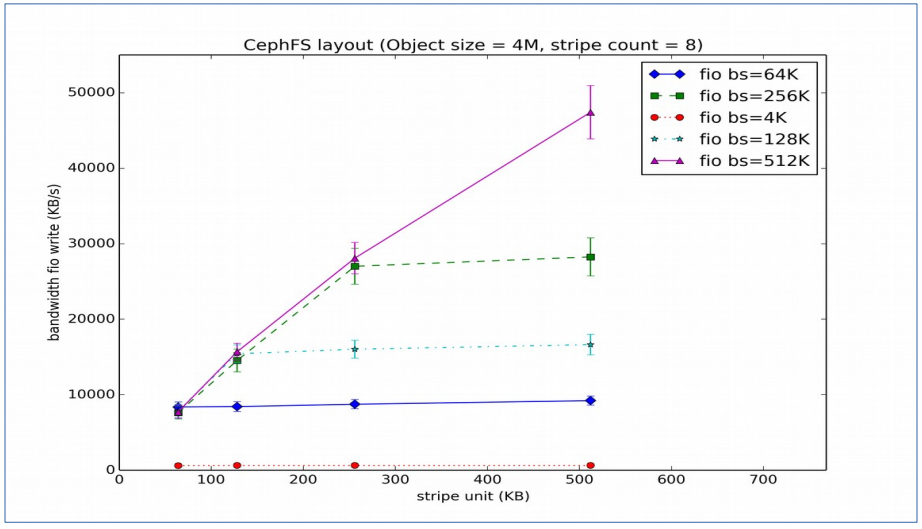
- **FIO bechmarks**
  - Sequential write and read, random write and read.
  - Files of 8 GB; ioengine=libaio; iodepth=64; direct=1.
  - Client cache purged before each test
  - More interested in understand CephFS layouts and File striping

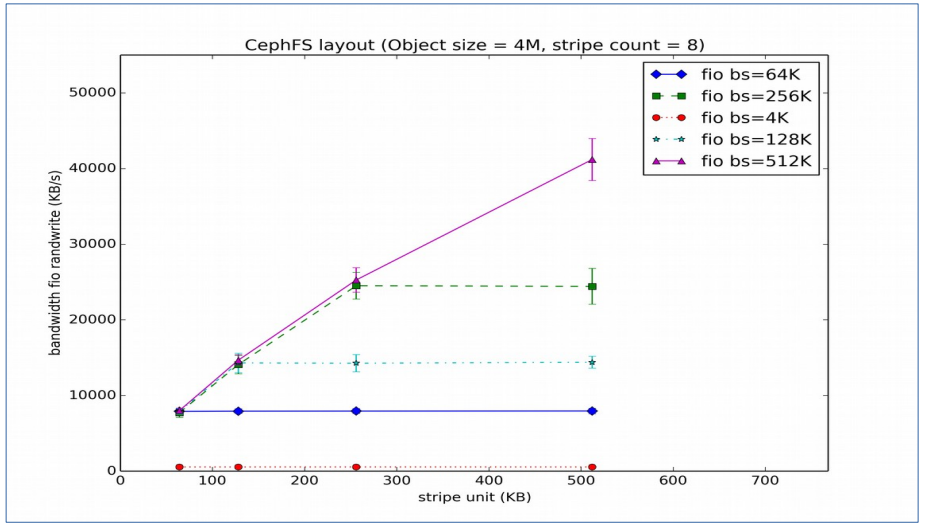
CephFS pre-production tests

- **Single FIO bechmark (stripe unit effect)**



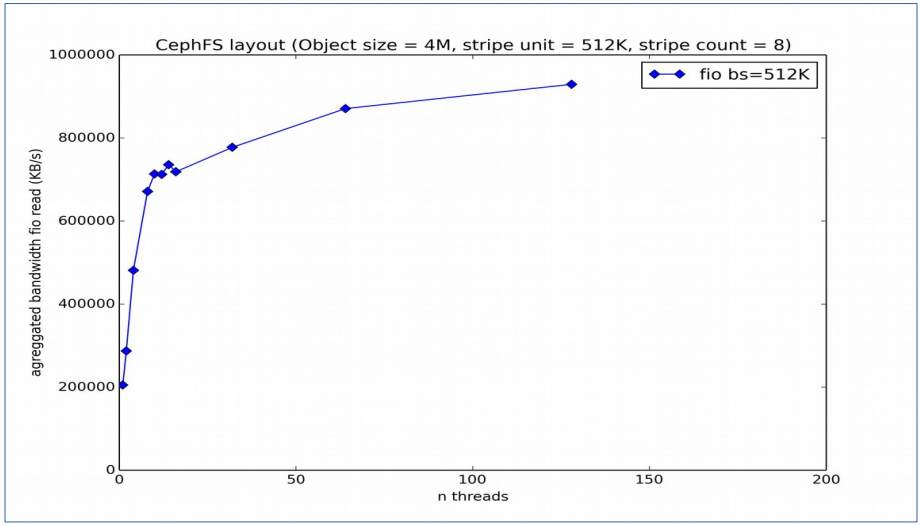


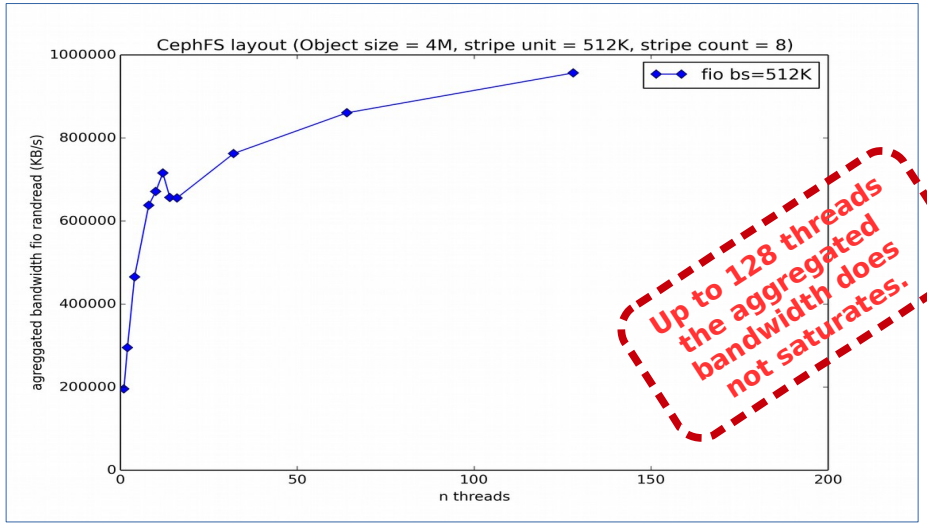


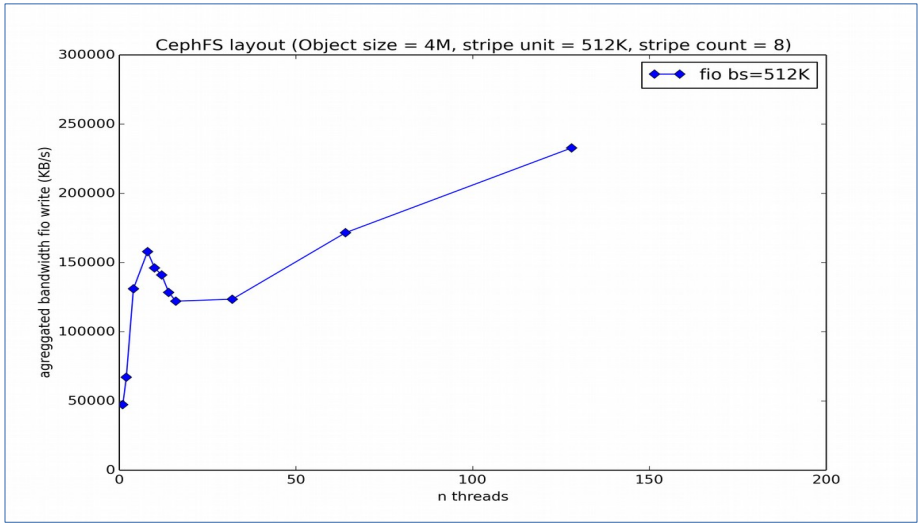


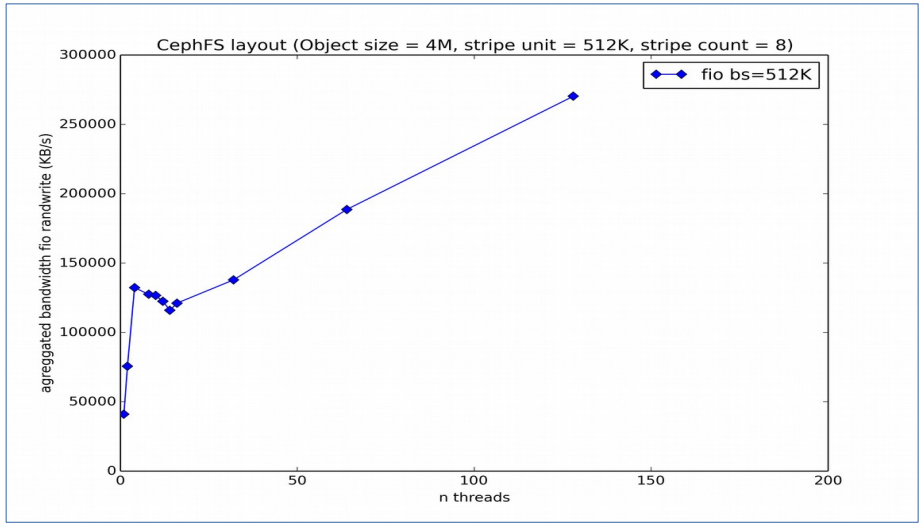
CephFS pre-production tests

- **Multithread FIO bechmark**







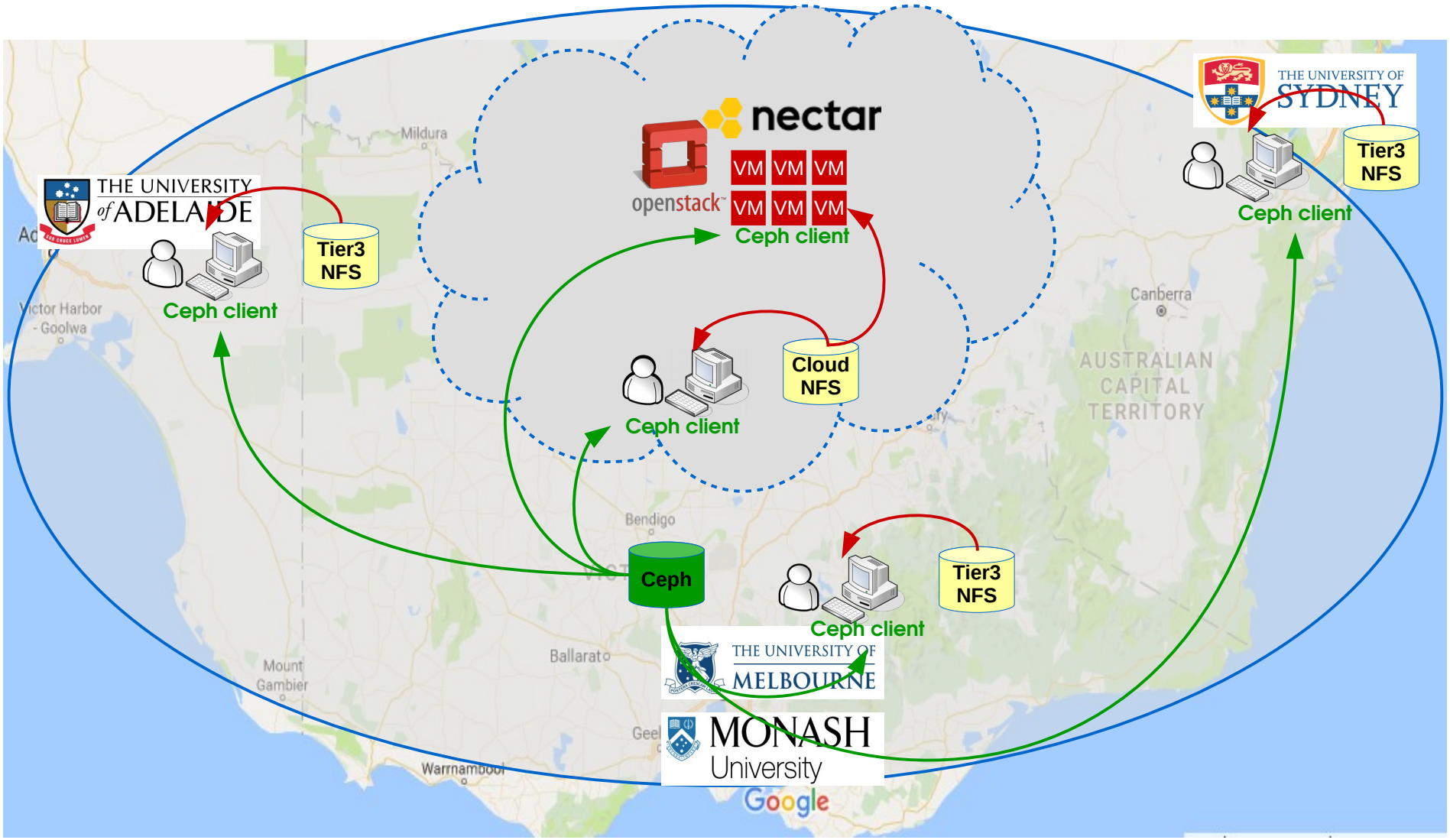


CephFS production hardware

- **Ceph Object Storage Cluster**
  - 3 monitors (physical hardware)
  - 11 storage servers, 112 OSD, 305 TB of raw storage
    - Centos7, 4 GB of RAM / OSD
    - 7 Dell PowerEdge R620 storage servers
      - PERC H710 Mini internal controller, PERC H810 external controllers
      - 4 storage servers x 8 OSDs (3 TB/each) + 3 storage servers x 16 OSD (3 TB /each)
    - 4 Dell PowerEdge R710
      - PERC 6/i internal controller, IBM Server RAID M5025 external controller
      - 8 OSD (3 TB / each) per storage server
    - Storage network with MTU 9000, txqueuelen + TX/RX buffer tuning
    - Intel DC S3550 SSDs (120 GB) for OSDs journals (4 OSDS : 1 SSD)

- **CephFS**
  - Dedicated pools for CephFS data and metadata; size=3, min_size=2 (3 replicas)
  - Active MDS → Dell PowerEdge R520, 32 GB RAM + 8 GB SWAP
  - Standby-Replay MDS → VM 8 GB RAM + 8 GB SWAP

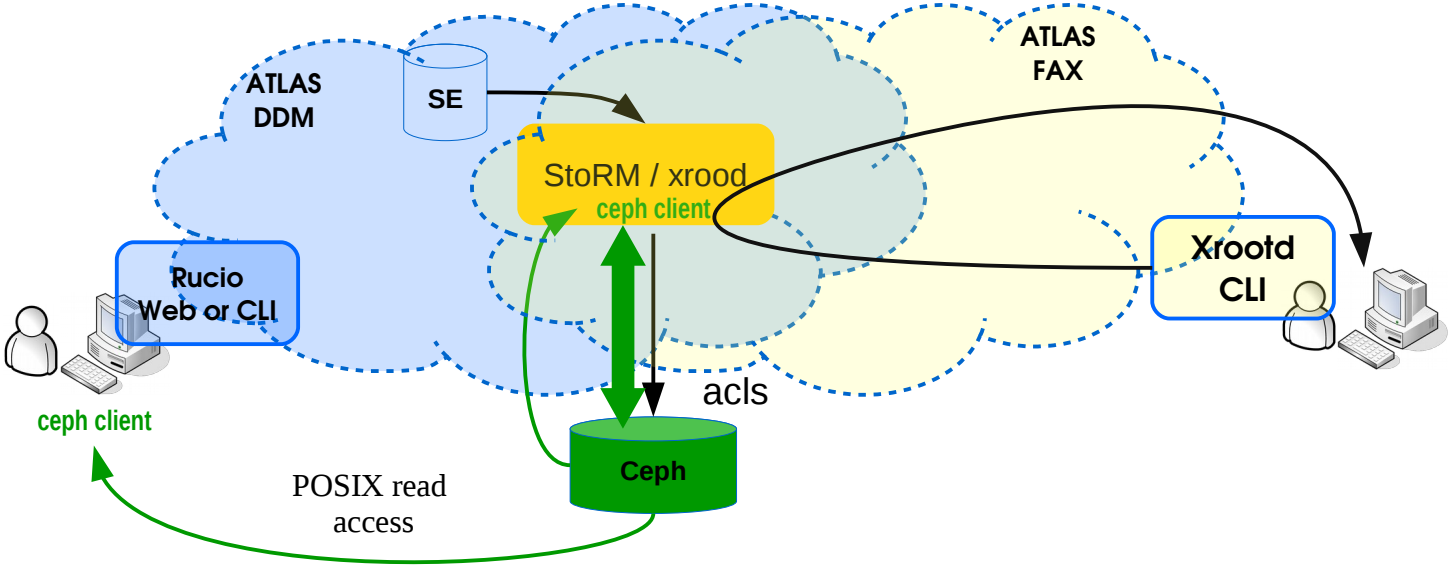
Enhanced 'Tier-3' services



Tier-2 integration (StoRM / xrootd)

- **ATLAS DDM integration**
  - StoRM (CephFS as data backend; Allows to set group ACLs on the fly)
  - A secondary ATLAS LOCALGROUP DISK served by cephfs (using the kernel client)
  - Read access for local atlas users

- **ATLAS FAX integration**
  - Xrootd server (using its posix interface)



A user perspective

- **CephFS is in 'Production' for more than an year**
  - Started with Infernalis and now in Jewel
  - Researchers are heavily using it. Apart from some minor issues (from a user's perspective), researchers are...

HAPPY!

HEPiX

Fuse vs Kernel client

- **CephFS clients (fuse vs kernel)**
  - The kernel client provides the best performance. However, it is always outdated in terms of bug fixes and enhanced functionalities.
  - The kernel client is not a flexible solution for our current OS distributions
    - Default kernels in Centos7 is 3.10 (too old in what regards ceph)
    - We would need to manually install ml / lt kernels from elrepo.
  - We opted for the fuse client because
    - Flexibility: Works in user space
    - Reliability: Synced with latest developments
    - Portability: Easily patched, recompiled and deployed.

- **CephFS (10.2.2) under SL6 .**
  - No support for RH6 flavours because it relies on C++11 features only available in GCC > 4.7. SL6 default GCC version is 4.4
  - Compile ceph in SL6 with GCC 4.8, Python 2.7, Fuse 2.9.2 and Boost 1.53
  - ceph-fuse Started by puppet and enabled via Environment Modules.
  - Normally running 200 ceph-fuse clients, mostly over WAN, with the potential to scale up once more VMs are started on Nectar cloud to satisfy demand.

Fuse (mount) options

- **Ceph-Fuse options are not always straight forward**

```
# ceph-fuse --id mount_user --fuse_default_permissions=0 --client_acl_type=posix_acl -k
/etc/ceph/ceph.client.mount_user.keyring -m <MON_IP>:6789 -r /cephfs /coepp/cephfs
```

- '- - id <user> -k /etc/ceph/<cluster_name>.client.<user>.keyring':
  - Mount cephfs using a specific / more restrict user/key

```
# cat /etc/ceph/ceph.client.mount_user.keyring
(client.mount_user
        key = ...
        caps mds = "allow"
        caps mon = "allow r"
        caps osd = "allow rw pool=coepp_cephfs_data"
```

- '--fuse_default_permissions=0 –client_acl_type=posix_acl':
  - Set acl support on ceph-fuse (only available in Jewel)
  - fuse kernel does not have ACL support. The '--fuse_default_permission=0' option disables kernel file permission check and let ceph-fuse do the check.

Fuse (mount) options

- **Ceph-Fuse options are not always straight forward**
  - '--client-quota' (not I mplemented in our case)
    - Enforce quotas (quotas are only available on ceph-fuse)
  - Possible to mount only a branch of the hierarchy

    # ceph-fuse --id mount_user --fuse_default_permissions=0 --client_acl_type=posix_acl -k /etc/ceph/ceph.client.mount_user.keyring -m <MON_IP>:6789 ***-r /coepp/cephfs /coepp/cephfs/syd***

- **Ceph-Fuse Caches**
  - Using ceph-fuse, there are two caches in play, one is in ceph-fuse, another one is the kernel pagecache.
  - When multiple clients read/write a file at the same time, ceph-fuse needs to disable cache and let reads/writes go to OSDs directly.
  - ceph-fuse can disable its own cache, but there is no way to disable the kernel pagecache dynamically. So client may read stale data from the kernel pagecache if that option is not in place.
  - Set 'fuse_disable_pagecache = true' in ceph.conf config file

Fuse virtual memory footprint

- Ceph-fuse shows a huge value for Virtual memory

```
PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
18563 root    20   0 10.0g 328m 5724 S  4.0  0.7   1:38.00 ceph-fuse
```

- Release compiled against against glibc > 1.10.
  - glibc > 1.10 introduces per-thread memory pools called arenas with up to 8 memory pools per core in 64 bits machines.
  - Each memory pool can take up to 64 MB of address space.
  - Our host had 24 cores (because of hyperthreading):
    8 x 24 x 64 MB = 12288 MB of virtual memory.
  - Ceph-fuse does not actually use this amount if memory but, because of the default malloc behaviour, it request that address space.
  - Possible to adjust the number of thread pools using MALLOC_ARENA_MAX env var.
    - MALLOC_ARENA_MAX=4 (instead of 8) with no performance impact.
  - There are plans to start compiling ceph with tcmalloc

Fuse client tunings

- **Ceph-Fuse WAN clients are "sensitive" to network perturbations**
  - You need to monitor the state of your client which can be become 'stale'
  - You may need to 'kick' the client session for the client to reconnect

```
#  ceph daemon /var/run/ceph/ceph-client.mount_user.asok mds_sessions
{
   "sessions": [
        (...)
         "state": "stale"
   ],
}

# ceph daemon /var/run/ceph/ceph-client.mount_user.asok kick_stale_sessions
```

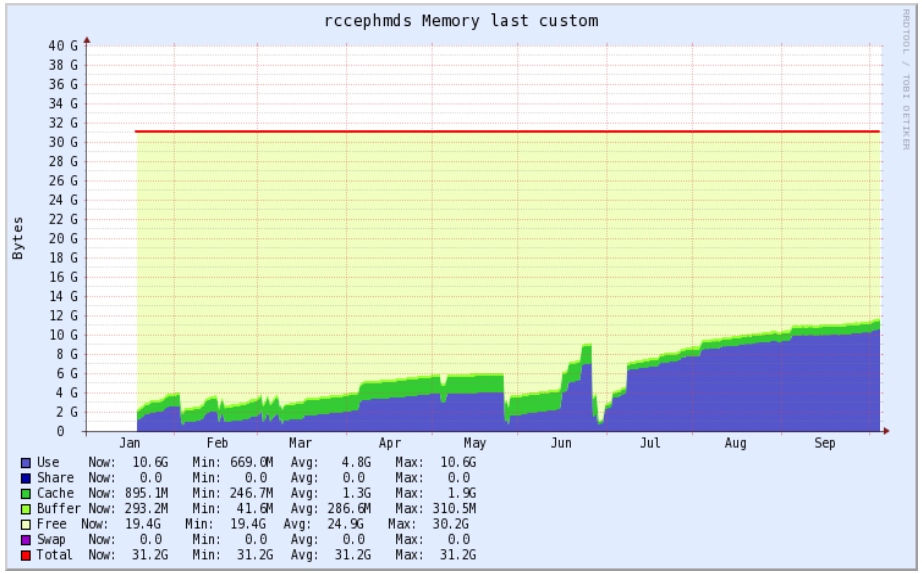
  - We may also want to play with specific timeout MDS settings
    - mds session timeout = 60, mds reconnect timeout = 45, mds session autoclose = 300

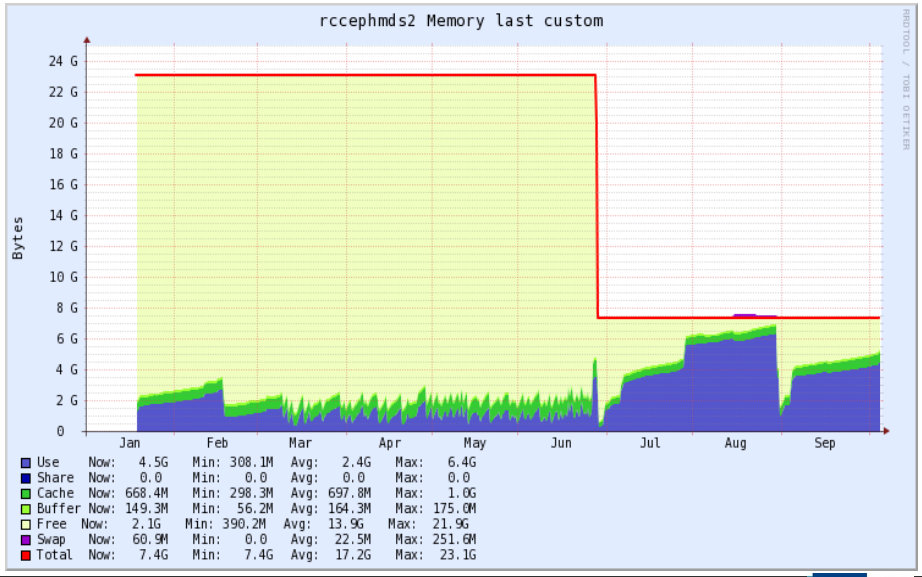
- **Prevent OOM kills**
  - ceph-fuse is provided via cvmfs (in our case)
  - When a user application misbehaves (takes large ammount of memory), cvmfs processes are one the first eligible processes to die and release memory.

Metadata server

- **Stable configuration (one active MDS + standby-replay MDS)**
  - Currently single threaded, SSD pools may help performance but not critical

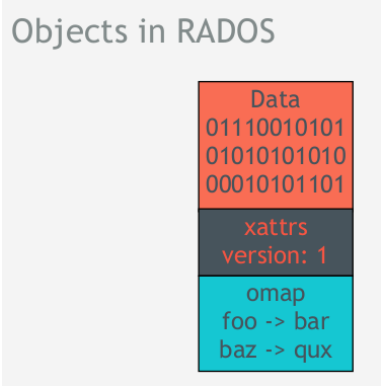
- **MDS is (mostly) about RAM**
  - You want to cache as many inodes as possible.
  - The default number of CInodes to cache is 100k. The size of metadata structures is:
    - CInode = 1400 bytes; CDentry = 400 bytes; CDir = 700 bytes → 2KB (?)
  - A 'back of an envelope' calculation give a way to low value 100k * 2KB ≃ 200 MB
  - You should increase 'mds cache size' if you have available RAM





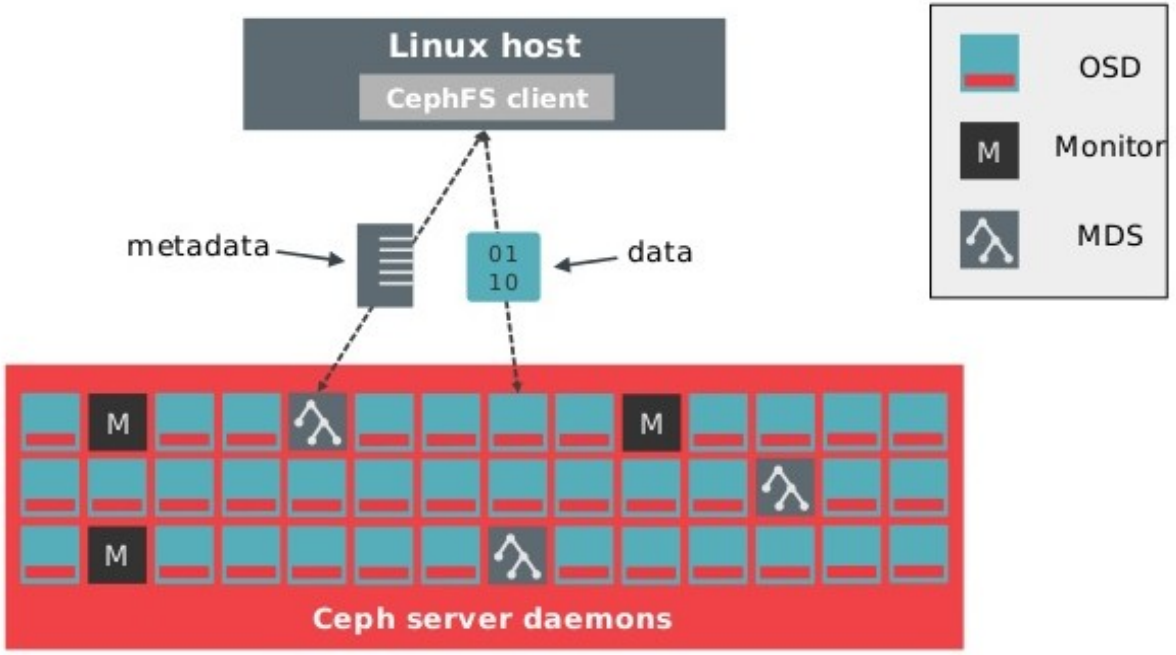
Metadata server

- **CephFS in RADOS**
  - 'User Data' is stored in RADOS OSDs as normal objects
  - 'User Data' metadata is stored as:
    - zero sized objects
    - Key-values in the OSDs leveldb (OMAP)



Zero size objects.

CephFS Metadata is stored as key-values in the osd leveldb (omap)



Issues: Metadata server

- **Listing directories with millions of files**
  - If metadata (omap key=values) > 2GB:
    - MDS wil hang because the metadata size is larger than max_message_size
    - Jewel 10.2.3 includes a limit on the number files per DIR to 100K.
    - Future releases will include UINT32 limit = (2^32)-1 (not ported to Jewel)
  - If metadata (omap key=values) < 2GB:
    - MDS will take a long, long time.
    - We're trying to read in a single 1-million-inode directory object. If the MDS cache size is too low, the MDS will immediately trims most of the directory out of cache.
    - The client submits a request for the next chunk, and it does it again, and again, ...
    - Assuming you have the RAM, by setting a larger "mds cache size" can minimize this behaviour.

- **Consider not trying to create directories that large if you don't need to**
  - there are plans for handling it much more gracefully in the future (in fact the code is one of the experimental features) but it's not enabled in Jewel.

Issues: Client / Metadata server

- **Client "failing to respond to cache pressure" issues**
  - This is a frequent message / issue in ceph that keeps popping up in the ML.
  - Clients maintain a metadata and data cache.
    - client_cache_size (default 16384) inodes of cached metadata. Items (such as inodes) in the client cache are also pinned in the MDS cache
    - When the MDS needs to shrink its cache (to stay within mds_cache_size), it sends messages to clients to shrink their caches too.
  - The client is unresponsive to MDS requests to release cached inodes
    - Either the client is unresponsive or has a bug

- **MDS out of memory**
  - If there is a client bug, this can prevent the MDS from properly staying within its mds_cache_size and it may eventually run out of memory and crash.
  - There is a confirmed bug in the kernel version (4.2.0) of this kind:
    - perf dump of MDS: "inode_max": 100000, "inodes": 4195342

CephFS Recover Tools

- **Complicated / Complex / Undocumented**
  - Use with extreme care…

- **Journal Issues / inconsistencies**
  - May prevent the write of metadata into RADOS.
  - MDS will keep trying to replay the journal
  - It is possible to try to flush whatever data is consistent to the object store
  - Alternatively, it should be possible to reset the journal which will result in losing the recent changes but preserves the filesystem.

```
--- *  backup journal  * ---
# cephfs-journal-tool journal export backup.bin

---* write any inodes/dentries recoverable from the journal into the backing store *---
--* (only if these inodes/dentries are higher-versioned than the previous contents of the backing store) *---
--* If any regions of the journal are missing/damaged, they will be skipped. *---
# cephfs-journal-tool event recover_dentries summary

---* Truncate the journal if it is really badly damaged *---
# cephfs-journal-tool journal reset
```

CephFS Recover Tools

- **Online / Forward scrubbing**
  - Happens in the background but can also be triggered manually

    # ceph mds mds.<id> scrub_path
    # ceph mds mds.<id> scrub_path recursive

  - Transverse the metadata and check if files (objects) are present. Only checks for the first object of a file. Will not cross check for the others.
  - As a result ceph may issue the message "Metadata data is damaged". This indicates that the damage was sufficiently isolated for the MDS to continue operating, although client accesses to the damaged subtree will return IO errors.

    --* provide more details on the damage*---
    # ceph tell mds.<id> damage ls

    --* will clear the damaged flag (doesn't fix anything, but it'll make the MDS try again *---
    # ceph mds repaired <rank>

CephFS Recover Tools

- **Offline / Backward scrubbing**
  - Transverse the objects space and try to regenerate metadata
  - A full offline scan_extents/scan_inodes run should re-link orphans into a top-level lost+found directory

---* scanning all objects to calculate size and mtime metadata for inodes *---
 # cephfs-data-scan scan_extents

–-* scanning the first object from every file to collect this metadata and inject it into the metadata pool*---
 # cephfs-data-scan scan_inodes

A site admin perspective

- **Managing CephFS...**
  - Deploying and installing is quick and easy; continuous operation is difficult. Problems are, most of time, only visible 'a posteriori.
  - Software (both Ceph and CephFS) a bit buggy
  - Some 'hairy' issues we already detected from a 'site admin' perspective
    - No real showstoppers but with some complexity involved

- **The disclaimer to our users is:**
  "Just put data there you can regenerate or retrieve from somewhere else"