Hi, I'm Greg

Greg Farnum
Ceph developer since 2009
Principal Software Engineer, Red Hat
gfarnum@redhat.com
Overview

- What is Ceph/Cephfs
- CephFS: What Works
  - It's a distributed POSIX filesystem!
  - There are many niceties that go with that
- CephFS: What Doesn't Work (Yet)
  - Directory fragmentation
  - Erasure Coding
  - Multi-Active MDS
  - Snapshots
- Pain Points & Use Cases
Where Does Ceph Come From?

- Then: UC Santa Cruz Storage Research Systems Center
- Long-term research project in petabyte-scale storage
- trying to develop a Lustre successor.

- Now: Red Hat, a commercial open-source software & support provider you might have heard of :) (Mirantis, SuSE, Canonical, 42on, Hastexo, ...)
- Building a business; customers in virtual block devices and object storage
- ...and reaching for filesystem users!
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP

RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift

RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver

CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE

NEARLY AWESOME

AWESOME

APP

HOST/VM

CLIENT

APP

HOST/VM

CLIENT

APP

HOST/VM

CLIENT

APP

HOST/VM

CLIENT

APP

HOST/VM

CLIENT
Now: Fully Awesome

**RGW**
- Web services gateway for object storage, compatible with S3 and Swift

**LIBRADOS**
- Client library allowing apps to access RADOS (C, C++, Java, Python, Ruby, PHP)

**RBD**
- Reliable, fully-distributed block device with cloud platform integration

**CEPHFS**
- Distributed file system with POSIX semantics and scale-out metadata management

**RADOS**
- Software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors
WHERE DO OBJECTS LIVE?
A METADATA SERVER?
CALCULATED PLACEMENT
CRUSH IS A QUICK CALCULATION
CRUSH AVOIDS FAILED DEVICES
CRUSH: DYNAMIC DATA PLACEMENT

CRUSH:
- Pseudo-random placement algorithm
  - Fast calculation, no lookup
  - Repeatable, deterministic
- Statistically uniform distribution
- Stable mapping
  - Limited data migration on change
- Rule-based configuration
  - Infrastructure topology aware
  - Adjustable replication
  - Weighting
DATA IS ORGANIZED INTO POOLS

POOL A
- 10 11 10 01
- 01 01 01 10

POOL B
- 01 10 11 10
- 10 01 01 01

POOL C
- 10 01 10 11

POOL D
- 01 10 01 01
- 11 10 01 10
- 01 01 10 01

CLUSTER
- POOLS (CONTAINING PGs)
RADOS COMPONENTS

**OSDs:**
- 10s to 10000s in a cluster
- One per disk (or one per SSD, RAID group...)
- Serve stored objects to clients
- Intelligently peer for replication & recovery

**Monitors:**
- Maintain cluster membership and state
- Provide consensus for distributed decision-making
- Small, odd number
- These do not serve stored objects to clients
RADOS: FAILURE RECOVERY

- Each OSDMap is numbered with an epoch number
- The Monitors and OSDs store a history of OSDMaps
- Using this history, an OSD which becomes a new member of a PG can deduce every OSD which could have received a write which it needs to know about
- The process of discovering the authoritative state of the objects stored in the PG by contacting old PG members is called **Peering**
RADOS: FAILURE RECOVERY

Epoch 20220:

11 → 5
LIBRADOS: RADOS ACCESS FOR APPS

LIBRADOS:

- Direct access to RADOS for applications
- C, C++, Python, PHP, Java, Erlang
- Direct access to storage nodes
- No HTTP overhead
- Rich object API
- Bytes, attributes, key/value data
- Partial overwrite of existing data
- Single-object compound atomic operations
- RADOS classes (stored procedures)
Existing Awesome Ceph Stuff
THE RADOS GATEWAY

APPLICATION

RADOSGW
LIBRADOS

socket

APPLICATION

RADOSGW
LIBRADOS

REST

socket

RADOS CLUSTER
RADOSGW MAKES RADOS WEBBY

RADOSGW:
- REST-based object storage proxy
- Uses RADOS to store objects
- API supports buckets, accounts
- Usage accounting for billing
- Compatible with S3 and Swift applications
STORING VIRTUAL DISKS

VM

HYPERSERVER

LIBRBD

RADOS CLUSTER

VM
RBD STORES VIRTUAL DISKS

RADOS BLOCK DEVICE:
- Storage of disk images in RADOS
- Decouples VMs from host
- Images are striped across the cluster (pool)
- Snapshots
- Copy-on-write clones
- Support in:
  - Mainline Linux Kernel (2.6.39+)
  - Qemu/KVM
  - OpenStack, CloudStack, Nebula, Proxmox
CephFS, The Awesome Parts
Awesomeness Timeline

1. **Pre-Awesome**
   - **Hammer** (LTS) - Spring 2015
   - **Infernalis** - Fall 2015

2. **Some Awesome**
   - **Jewel** (LTS) - Spring 2016
   - **Kraken** - Fall 2016

3. **More Awesome**
   - **Luminous** (LTS) - Spring 2017

Version Numbers:
- 0.94.z
- 10.2.z
- 12.2.z
Awesome: It's A Filesystem!
POSIX Filesystem

- Mounting, from multiple clients
  - Not much good without that!
- POSIX-y goodness:
  - Atomic updates
  - Files, with names and directories and rename
- Coherent caching
  - Updates from one node are visible elsewhere, immediately
LINUX HOST

KERNEL MODULE

Ceph-fuse, samba, Ganesha

metadata

01 10

data

RADOS CLUSTER
POSIX Filesystem: Consistency

- CephFS has “consistent caching”
- Clients are allowed to cache, and the server invalidates them before making changes
  - This means clients never see stale data of any kind!
  - And there's no opportunity for any kind of split brain situation
POSIX Filesystem: Scaling Data

- All data is stored in RADOS
- Filesystem clients write directly to RADOS
- Need more data space? Add more OSDs!
- Faster throughput?
  - Faster SSDs!
  - Wider striping of files across objects!
  - ...at least, up until you're limited by latency instead of throughput
POSIX Filesystem: Scaling Metadata

- Only **active** metadata is stored in memory
- Size your metadata server (MDS) by active set size, not total metadata
rstats are cool

# ext4 reports dirs as 4K
ls -lhd /ext4/data

```
drwxrwxr-x. 2 john john 4.0K Jun 25 14:58 
/home/john/data
```

# cephfs reports dir size from contents
$ ls -lhd /cephfs/mydata

```
drwxrwxr-x. 1 john john 16M Jun 25 14:57 ./mydata
```
Awesome: A Security Model
CephX security capabilities

- Clients start out unable to access the MDS.
  - Incrementally granted permissions for subtrees (or the whole tree)
  - To act as a specific user
  - Etc
- For real security, these must be coordinated with OSD caps:

```
ceph auth get-or-create client.foo
  mds "allow rw path=/foodir"
  osd "allow rw pool=foopool"
  mon "allow r"
```
CephX security capabilities: Protection

- The security capabilities are encrypted by the server; can't be changed by client
- MDS only examines MDS grants
  - Protects against acting as an unauthorized user
  - Prevents all access to inodes/dentries not under granted path
- OSDs independently examine OSD grants
  - Protects against access to unauthorized pools and namespaces

- Possible hole: if clients share namespace+pool, they can trample on raw file data
  - If you don't trust your clients, give them each their own namespace (free for RADOS) and specify it in CephFS layout for their directory hierarchy
Awesome: Hot standby MDS
Standby servers

- Nothing ties metadata to a particular server!

- Spin up an arbitrary number of “standby” and “standby-replay” servers
  - Standby: just waiting around; can be made active
  - Standby-replay: actively replaying the MDS log

- Warms up the cache for fast takeover

```
rename /tmp/file1 -> /home/greg/foo
rename /tmp/file2 -> /home/greg/bar
create /home/greg/baz
```
Standby servers: reconnect

- Replay log, load all necessary file data from RADOS
- Let clients replay uncommitted operations, process them
- Synchronize caching states between clients and MDS
- Go active!
Mostly Awesome: Scrub/Repair
Forward Scrub

- Forward scrubbing, to ensure consistency
  
  ceph daemon mds.<id> scrub_path
  ceph daemon mds.<id> scrub_path recursive
  ceph daemon mds.<id> scrub_path repair
  ceph daemon mds.<id> tag path

- You have to run this manually right now, no automatic background scrub :(
  - Fix: targeted for Luminous! With multi-MDS support!
Repair tools: cephfs-journal-tool

• Disaster recovery for damaged journals:
  – inspect/import/export/reset
  – header get/set
  – event recover_dentries

• Allows rebuild of metadata that exists in journal but is lost on disk

• Companion cephfs-table-tool exists for resetting session/inode/snap tables as needed afterwards.
Repair tools: cephfs-data-scan

• “Backwards scrub”

• Iterate through all RADOS objects and tie them back to the namespace

• Parallel workers, thanks to new RADOS functionality
  - cephfs-data-scan scan_extents
  - cephfs-data-scan scan_inodes
Repair tool methods

- Examine object names and send inferred stat info to "root" object

1000.1

1000.0

/v2/home/v5/greg/v9/foo
Repair tool methods

- Assemble tree information from backtrace and inferred stat

```
/mydir, total size 2074 bytes

foo -> ino 1342, 6 MB
bar -> ino 1001, 1024 bytes
baz -> ino 1242, 2 MB
```

```
/v2/home/v5/greg/v9/foo
```

```
<ino 0,v2>/home<ino 1,v5>/greg<ino 5,v9>/
```

```
1342.0
```
Repair tool methods

- Do inference and then insertion in parallel across the cluster
CephFS: The Parts You Don't Get
Directory Fragmentation

- Directories are generally loaded from disk as a unit
  - But sometimes that's too much data at once!
  - Or you want to spread a hot directory over many active MDSes

<table>
<thead>
<tr>
<th>Path</th>
<th>Directory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ino 0,v2&gt;/home&lt;ino 1,v5&gt;/greg&lt;ino 5,v9&gt;/</td>
<td>Mydir[01], total size 7MB</td>
</tr>
<tr>
<td>foo -&gt; ino 1342, 4 MB</td>
<td></td>
</tr>
<tr>
<td>bar -&gt; ino 1001, 1024 KBytes</td>
<td></td>
</tr>
<tr>
<td>baz -&gt; ino 1242, 2 MB</td>
<td></td>
</tr>
<tr>
<td>&lt;ino 0,v2&gt;/home&lt;ino 1,v5&gt;/greg&lt;ino 5,v9&gt;/</td>
<td>Mydir[10], total size 8MB</td>
</tr>
<tr>
<td>hi -&gt; ino 1000, 6 MB</td>
<td></td>
</tr>
<tr>
<td>hello -&gt; ino 6743, 1024 KB</td>
<td></td>
</tr>
<tr>
<td>whaddup -&gt; ino 9872, 1 MB</td>
<td></td>
</tr>
</tbody>
</table>
Directory Fragmentation: What's Short

- It's not well-tested
  - Just need to do the QA work
  - Expected in Luminous

<table>
<thead>
<tr>
<th>Directory 1</th>
<th>Directory 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>/&lt;ino 0,v2&gt;/home&lt;ino 1,v5&gt;/greg&lt;ino 5,v9&gt;/</td>
<td>/&lt;ino 0,v2&gt;/home&lt;ino 1,v5&gt;/greg&lt;ino 5,v9&gt;/</td>
</tr>
<tr>
<td>Mydir[01], total size 7MB</td>
<td>Mydir[10], total size 8MB</td>
</tr>
<tr>
<td>foo -&gt; ino 1342, 4 MB</td>
<td>hi -&gt; ino 1000, 6 MB</td>
</tr>
<tr>
<td>bar -&gt; ino 1001, 1024 KBytes</td>
<td>hello -&gt; ino 6743, 1024 KB</td>
</tr>
<tr>
<td>baz -&gt; ino 1242, 2 MB</td>
<td>whaddup -&gt; ino 9872, 1 MB</td>
</tr>
</tbody>
</table>
Almost Awesome: Active Multi-MDS
Active Multi-MDS

- Because no metadata is stored on MDS servers, migrating it is “easy”!
Active Multi-MDS

Cooperative Partitioning between servers:

• Keep track of how hot metadata is
• Migrate subtrees to keep heat distribution similar
  – Cheap because all metadata is in RADOS
• Maintains locality
Active Multi-MDS: What's short

- MDS failure/recovery in general is more complicated with >1 active MDS
  - The coding is detailed and takes time to get right
- Testing
- Targeted for Luminous
  ...but we’ll see
Almost Awesome: Snapshots
Snapshots: Disk Data Structures

- Arbitrary sub-tree snapshots of the hierarchy
- Metadata stored as old_inode_t map in memory/disk
- Data stored in RADOS object snapshots

```
<ino 0,v2>/home<ino 1,v5>/greg<ino 5,v9>/
Mydir[01], total size 7MB
foo -> ino 1342, 4 MB, [<1>,<3>,<10>]
bar -> ino 1001, 1024 KBytes
baz -> ino 1242, 2 MB
```

```
<v2>/home<v5>/greg<v9>/foo
```

```
1342.0
```
Snapshots

- Arbitrary sub-tree snapshots of the hierarchy
- Metadata stored as old_inode_t map in memory/disk
- Data stored in RADOS object snapshots

1342.0/1
/v1/home/v3/greg/v7/foo

1342.0/HEAD
/v2/home/v5/greg/v9/foo
Snapshots: What's Short

• Testing. So much testing.
• The exciting combinatorial explosion of tracking all this across different metadata servers!
  – Much of this exists; it's incomplete in various ways
  – As always, recovering from other failures which impact our state transitions
• Targeted for after Luminous
  – It works pretty well on single-MDS systems, but that’s boring
Almost Awesome: Multi-FS
MultiFS: What's Present

- You can create multiple filesystems within a RADOS cluster
  - Different pools or namespaces
- Each FS gets its own MDS and has to be connected to independently
MultiFS: What's Missing

- Testing: This gets limited coverage in our test suite
- Security model: we know where we're going, but it's not done
  - Can't expose filesystem existence to users who aren't allowed to see it
- Post-luminous
Pain Point: File Deletion
File Deletion

- The MDS deletes RADOS objects in the background after files are unlinked
- This requires “pinning” the inode in memory
- Usually not a problem, unless you have so many deleting files your MDS memory cache fills up!
File Deletion: The Fix

- Pull request pending: build a queueing system in RADOS
  - https://github.com/ceph/ceph/pull/12786
  - Add files to delete queue
  - Pull them off and delete, in constant memory space
- This will be done for Luminous
Pain Point: Client Trust
Client Trust

- Clients can trash anything they can write
  - Give clients separate namespaces!
- Clients can deny writes to anything they can read
  - Don't share stuff across tenants
- Clients can DoS the MDS they attach to
  - ...Multiple FSes in a cluster will fix this

- This is pretty fundamental. If you actively don’t trust your clients, put them behind an NFS gateway.
Pain Point: Debugging Live Systems
Exposing State: What's available

ceph daemon mds.a dump_ops_in_flight
{
    "ops": [
        {
            "description": "client_request(client.
            "initiated_at": "2015-03-10 22:26:17.4
            "age": 0.052026,
            "duration": 0.001098,
            "type_data": ["submit entry: journal_and_reply",
            "client.4119:21120",
            ...
        },
        ...
Exposing State: What's available

# ceph daemon mds.a session ls
...
  "client_metadata": {
    "ceph_sha1": "a19f92cf...",
    "ceph_version": "ceph version 0.93...",
    "entity_id": "admin",
    "hostname": "claystone",
    "mount_point": "\home\john\mnt"
  }

Exposing State: What's available

- ceph mds tell 0 dumpcache /path/to/dump/to

- Yes, it seriously dumps the full cache to a file
Exposing State: What's missing

- Dumping individual dirs/dentries/inodes
- Good ways of identifying why things are blocked
- Tracking accesses to a file
- ...and other things we haven't thought of yet?
Erasure Coding (Overwrites)

- Instead of replicating across OSDs, give them shards and parity blocks
- Current EC RADOS pools are append only
  - simple, stable suitable for RGW, or behind a cache tier
- Coming in Luminous: EC with overwrite support
  - This will be slow at first, as it requires a two-phase commit and optimizations to be remotely efficient will follow
  - Means you can store CephFS and RBD data at 1.5x instead of 3x cost, with same (or larger) number of node failures
• BlueStore = Block + NewStore
  - key/value database (RocksDB) for metadata
  - all data written directly to raw block device(s)
  - can combine HDD, SSD, NVMe, NVRAM
• Full data checksums (crc32c, xxhash)
• Inline compression (zlib, snappy, zstd)
• ~2x faster than FileStore
  - better parallelism, efficiency on fast devices
  - no double writes for data
  - performs well with very small SSD journals
• New implementation of network layer
  – replaces aging SimpleMessenger
  – fixed size thread pool (vs 2 threads per socket)
  – scales better to larger clusters
  – more healthy relationship with tcmalloc
  – now the default!

• Pluggable backends
  – PosixStack – Linux sockets, TCP (default, supported)
  – Two experimental backends!
• ceph-mon monitor daemons currently do a lot
  – more than they need to (PG stats to support things like 'df')
  – this limits cluster scalability
• ceph-mgr moves non-critical metrics into a separate daemon
  – that is more efficient
  – that can stream to graphite, influxdb
  – that can efficiently integrate with external modules (even Python!)
• Good host for
  – integrations, like Calamari REST API endpoint
  – coming features like 'ceph top' or 'rbd top'
  – high-level management functions and policy

(time for new iconography)
Who Should Use CephFS?
CephFS Users

- Some vendors are targeting it at OpenStack users, since it pairs so well with RBD
- CephFS is good at large files
  - It does well with small files for a distributed FS, but there’s no comparing to a local FS
- CephFS for home directories?
  - If your users are patient or metadata can all be cached, but remember you want very new clients to get all the bug fixes
- Anybody who likes exploring: go for it!
• docs
  – http://docs.ceph.com/
  – https://github.com/ceph

• help
  – ceph-users@ceph.com, ceph-devel@vger.kernel.org
  – #ceph, #ceph-devel on irc.oftc.net
THANK YOU!

Greg Farnum

gfarnum@redhat.com
@gregsfortytwo