redhat.

Large-scale Stability and Performance of the Ceph File System

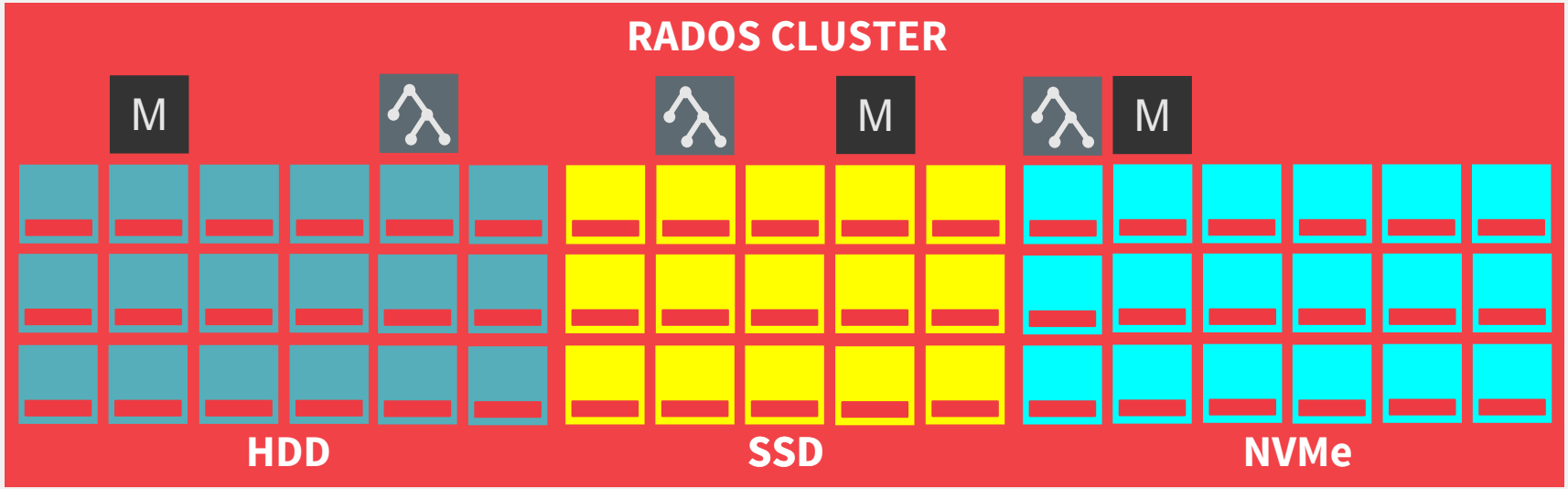
Vault 2017

Patrick Donnelly
Software Engineer
2017 March 22

Introduction to Ceph

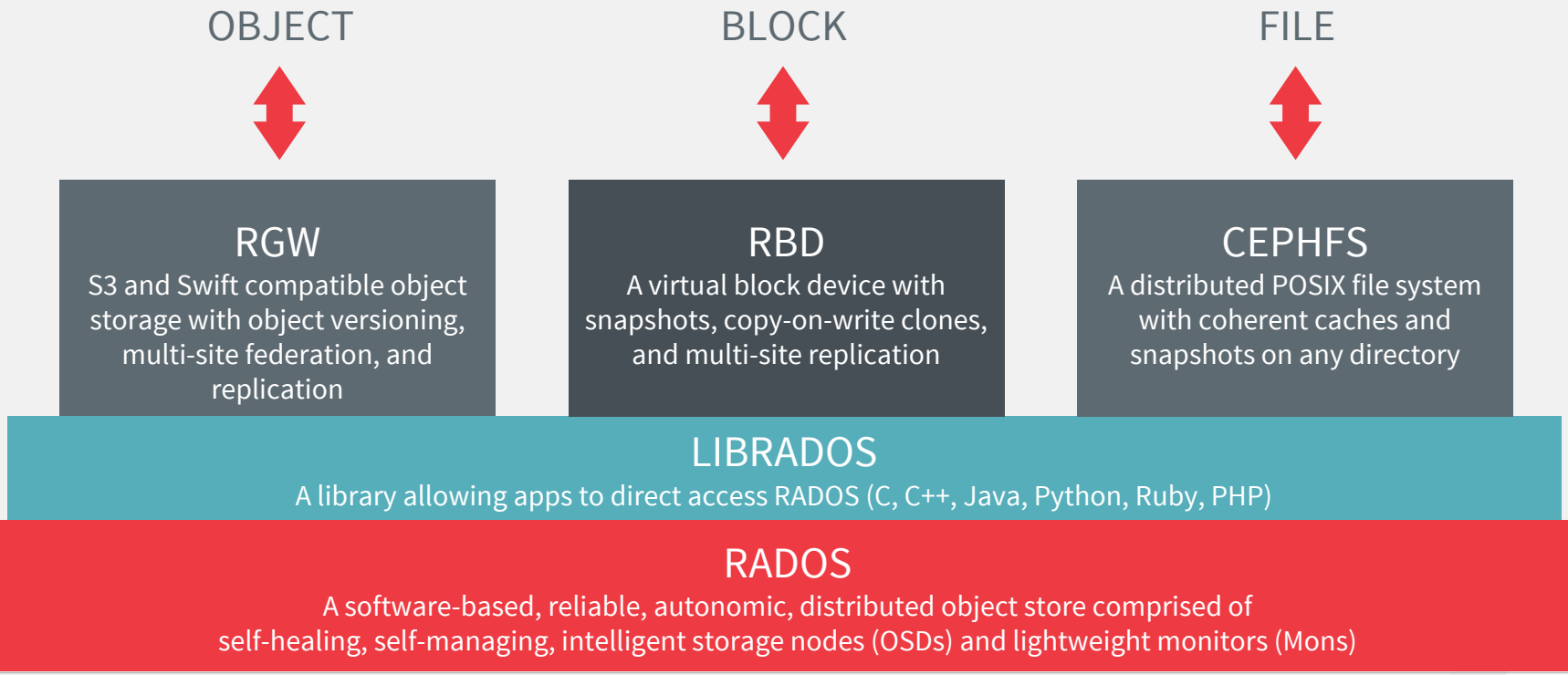
- Distributed storage
  All components scale horizontally
  No single point of failure
  Software
  Hardware agnostic, commodity hardware
  Object, block, and file in a single cluster
  Self-manage whenever possible
  Open source (LGPL)

Ceph's RADOS: Reliable Autonomic Distributed Object Store



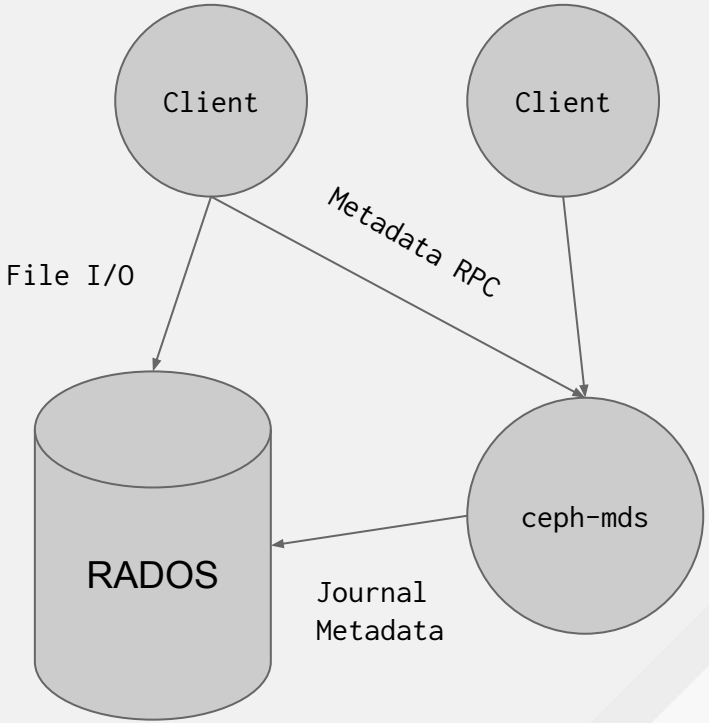
- Single Ceph cluster
- Separate RADOS pools for different storage hardware types

Ceph Components



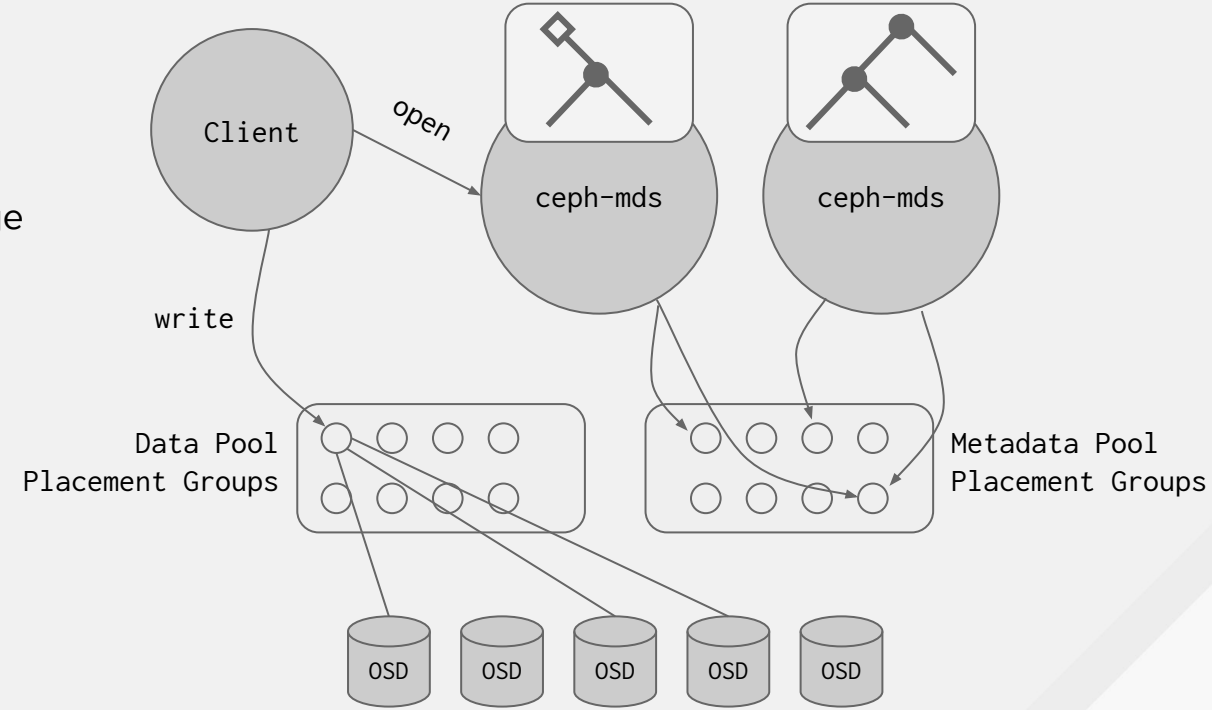
What is CephFS? Why use it?

- CephFS is a POSIX-compatible distributed file system!
- File based workloads.
- Managed and hierarchical shared workspaces.
  - Openstack Manila shares
- Coherent caching across clients.
  - Synchronous updates visible everywhere.
- All (meta)data stored in RADOS.
- Clients access data directly via RADOS.



CephFS Architecture

- MDS run on RADOS
  - No host level storage
  - Serves as cache for metadata
- CRUSH+RADOS used to place file data and metadata.



Fundamental Concepts in CephFS

- Clients can **locate** and read/write file data autonomously.
- MDS only manages file system metadata: `mkdir`, `unlink`, `open`, etc.
- Horizontal scaling through dynamic subtree partitioning (enabled by embedded inodes)
  - Directories authoritatively managed by one of many MDS
  - Hotspots resolved by splitting trees (and even directories)

Performance Testing

Why use the cloud for performance testing?

- Rapid scale!
- Costs proportional to the amount of time you want to use the hardware
- Run your test and tear down the cluster
- Concurrent tests on the same hardware configuration

Cloud Deployment using Linode

- Low cost VPS provider
- Smaller plans available perfect for micro-services (like the monitors and even the OSDs)
- API available for rapid automated deployment

Issues:

- Vagrant launching of Linodes is horribly slow
- API rate throttling (fixed now!)

Solution: use API-call efficient python script to do what you need

`https://github.com/batrick/ceph-ansible/blob/multimds-tests/linode-launch.py`

Hardware

Plan	CPU	RAM	SSD	Cost
1024	1	1GB	20GB	.0075$/hr
2048	1	2GB	30GB	.0150$/hr
4096	2	4GB	48GB	.0300$/hr
8192	4	8GB	96GB	.0600$/hr

Costs

1TB usable for Ceph

Daemon	Count	Machine Plan	Rate	Cost
mon	3	1024	.0075$/hr	.0225$/hr
osd	64	1024	.0075$/hr	.4800$/hr
mds	16	8192	.06$/hr	.9600$/hr
client	64	4096	.03$/hr	1.9200$/hr
	147			3.4025$/hr

Ceph Cluster Performance in the Cloud!

This terminal output has been modified from its original version. It has been formatted to fit this screen.

Performance: Transfers

```
$ iperf -c 192.168.192.238 -i1 -t 10
Client connecting to 192.168.192.238, TCP port 5001
TCP window size: 45.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.222.186 port 60814 connected with 192.168.192.238 port 5001
[ ID] Interval       Transfer   Bandwidth
[  3]  0.0- 1.0 sec   243 MBytes  2.04 Gbits/sec
[  3]  1.0- 2.0 sec   113 MBytes   948 Mbits/sec
[  3]  2.0- 3.0 sec   103 MBytes   866 Mbits/sec
```

Performance: Transfers

```
[root@li234-130 ~]# dd if=/dev/zero of=foo bs=64k count=16384 oflag=direct
1073741824 bytes (1.1 GB) copied, 2.98743 s, 359 MB/s
[root@li234-130 ~]# dd if=/dev/zero of=foo bs=64k count=16384 oflag=direct
1073741824 bytes (1.1 GB) copied, 2.8166 s, 381 MB/s
[root@li234-130 ~]# dd if=/dev/zero of=foo bs=64k count=16384 oflag=direct
1073741824 bytes (1.1 GB) copied, 3.13887 s, 342 MB/s
```

Performance: Client Object Writes

Not that uncommon even on dedicated hardware:
https://www.sebastien-han.fr/blog/2012/08/26/ceph-benchmarks/

```
$ rados bench -p cephfs_data --write-object 300 write --no-cleanup
Maintaining 16 concurrent writes of 4194304 bytes to objects of size 4194304...
…
Total time run:       300.497885 Total writes made:       8938
Write size:           4194304    Object size:             4194304
Bandwidth (MB/sec):   118.976    Stddev Bandwidth:        8.67456
Max bandwidth (MB/sec): 208      Min bandwidth (MB/sec): 100
Average IOPS:         29         Stddev IOPS:             2
Max IOPS:             52         Min IOPS:                25
Average Latency(s):   0.537848   Stddev Latency(s):       0.43588
Max latency(s):       2.81207    Min latency(s):          0.0370765
```

Performance: Client Object Sequential Reads

```
$ rados bench -p cephfs_data 300 seq
...
Total time run:       57.142655 Total reads made:     8938
Read size:            4194304   Object size:          4194304
Bandwidth (MB/sec):    625.662  Average IOPS:         156
Stddev IOPS:          6         Max IOPS:             168
Min IOPS:             140       Average Latency(s):    0.101394
Max latency(s):       0.673822  Min latency(s):       0.015606
```

Performance: Client Object Random Reads

```
$ rados bench -p cephfs_data 300 rand
...
Total time run:       300.116764 Total reads made:    47291
Read size:            4194304    Object size:         4194304
Bandwidth (MB/sec): 630.301    Average IOPS:         157
Stddev IOPS:          8          Max IOPS:            173
Min IOPS:             75         Average Latency(s):   0.100707
Max latency(s):       2.297      Min latency(s):      0.0150806
```

Performance: Client Object omap Write

```
$ rados bench -p cephfs_data --write-omap 300 write
...
Total time run:       300.533417 Total writes made:       8937
Write size:           4194304    Object size:             4194304
Bandwidth (MB/sec):   118.949    Stddev Bandwidth:        9.24203
Max bandwidth (MB/sec): 204      Min bandwidth (MB/sec): 92
Average IOPS:         29         Stddev IOPS:             2
Max IOPS:             51         Min IOPS:                23
Average Latency(s):   0.537945   Stddev Latency(s):       0.460546
Max latency(s):       11.585     Min latency(s):          0.0432125
```

Ceph Deployment with ceph-ansible

- Ansible automates cloud provisioning, configuration management, and application deployment.
- Deploys ceph to a cluster:
  - Installs repos and packages for a desired version (even `master`!)
  - Sets up keys/config files
  - Starts daemons on nodes
  - Configures cluster

ceph-ansible: Define cluster nodes

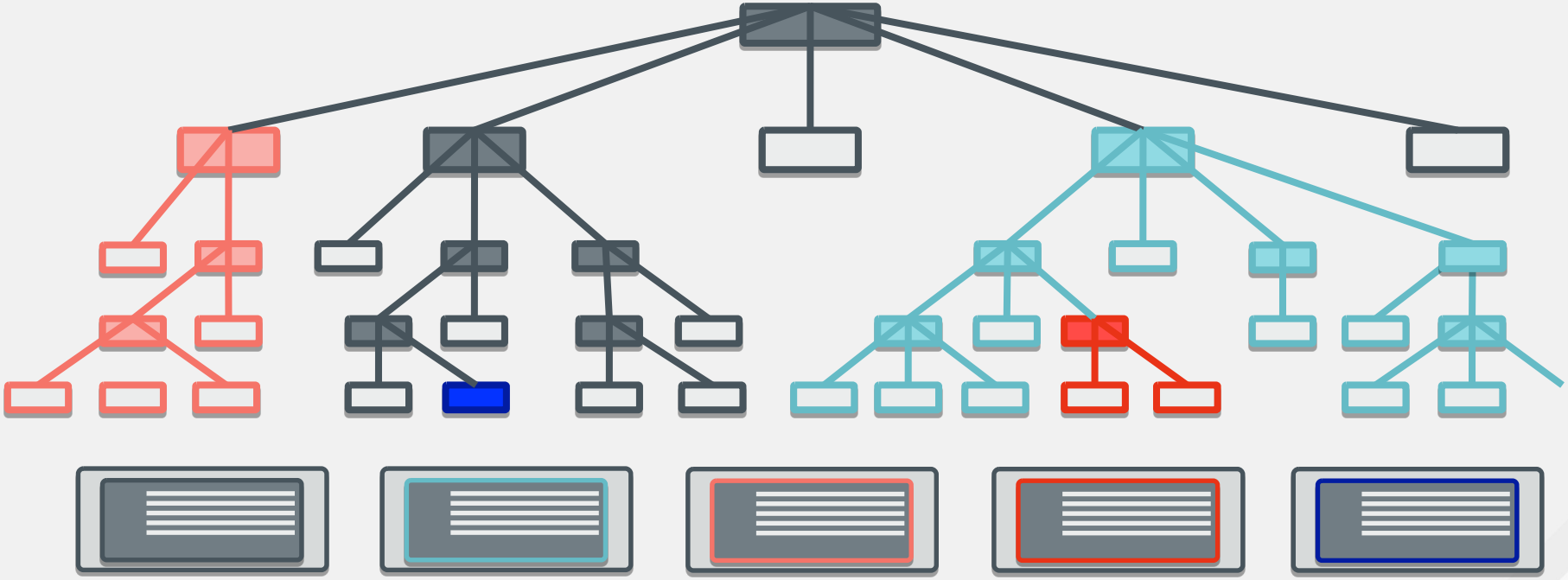
```
[mdss]
       ceph-mds-0 ansible_ssh_host=66.175.208.228 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
       ceph-mds-1 ansible_ssh_host=198.74.59.34 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
       ceph-mds-2 ansible_ssh_host=162.216.16.10 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
[clients]
       ceph-client-0 ansible_ssh_host=23.92.20.8 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
       ceph-client-1 ansible_ssh_host=23.92.16.205 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
…
[mons]
       ceph-mon-0 ansible_ssh_host=45.56.110.10 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
       ceph-mon-1 ansible_ssh_host=45.33.70.233 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
       ceph-mon-2 ansible_ssh_host=45.33.78.51 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
[osds]
       ceph-osd-0 ansible_ssh_host=45.33.81.105 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
       ceph-osd-1 ansible_ssh_host=45.33.89.87 ansible_ssh_port=22 ansible_ssh_user='root' ansible_ssh_private_key_file='id_rsa'
…
```

ceph-ansible: group_vars/all config

```
---

ceph_dev: true
ceph_dev_branch: wip-multimds-tests

pool_default_pg_num: 512
#pool_default_pg_num: 4096

ceph_conf_overrides:
  client:
       fuse default permissions: false
       log file: /var/log/ceph/ceph-client-$pid.log
       admin socket: /var/run/ceph/ceph-client-$pid.asok
  mds:
       debug mds: 10/20
       #mds cache size: 200000

mds_allow_multimds: true
mds_max_mds: 2
```

CephFS Multiple Metadata Server Performance

CephFS Subtree Partitioning

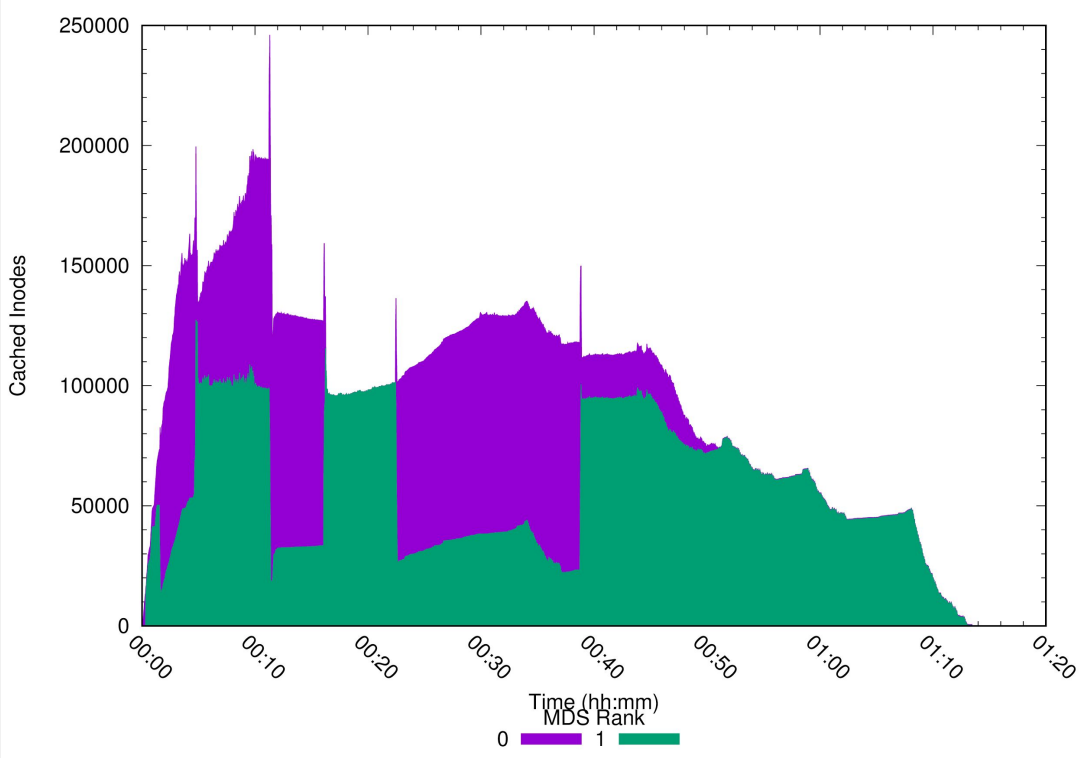


Balancing Subtrees

- Share load across multiple metadata servers
- Split subtrees into parts based on load
- Merge subtrees once load cools

Default Balancer

- Default balancer tracks each inode's popularity based on operations, total load is shared to inform balancing decisions
- However, early balancer experiments showed imbalance and volatility
- Experiment: 2 active MDS with 8 clients building the kernel independently



Stop-Gap: Balance by Pinning Subtrees

- Idea: allow the cluster admin to manually partition the subtrees by pinning a subtree to a particular MDS, preventing its export to another MDS

```
setfattr -n ceph.file.pin -v 2 path
```

- Ticket: http://tracker.ceph.com/issues/17834
- To evaluate effectiveness and usefulness, a manual balancer is used in experiments!

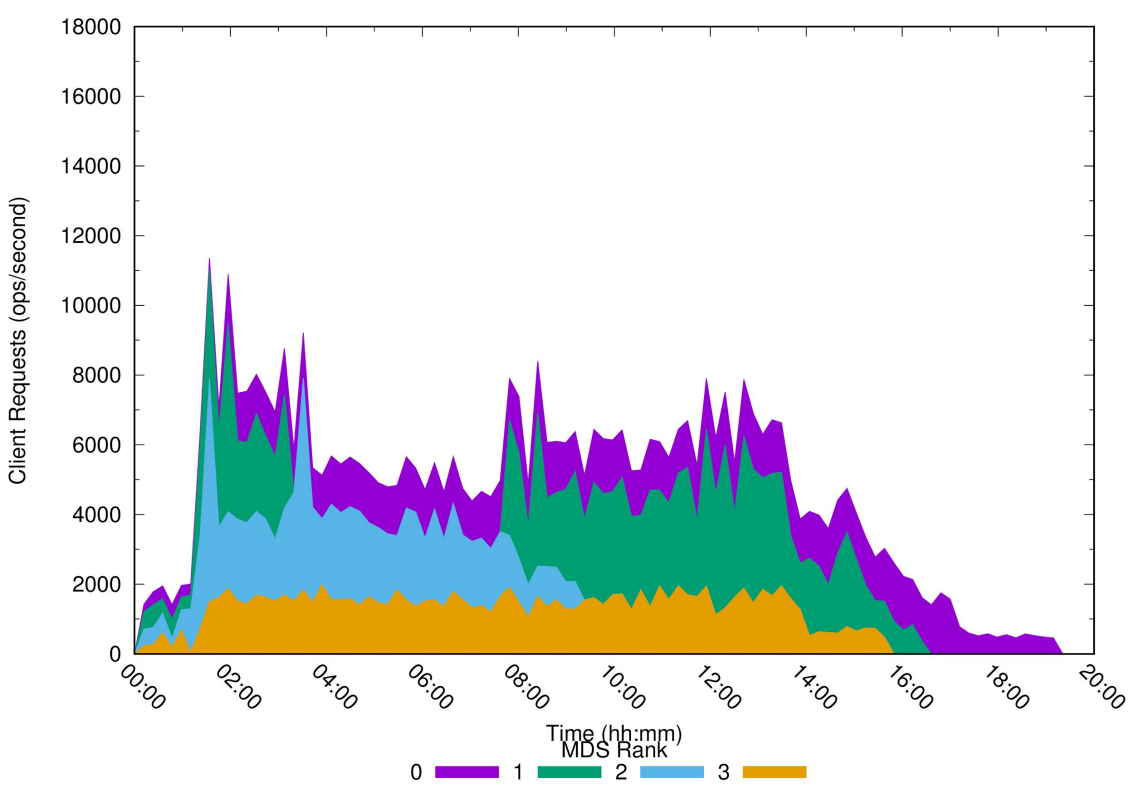
Experimental Setup: kernel clients

An old file system test: build the kernel! (Then ship the product!)

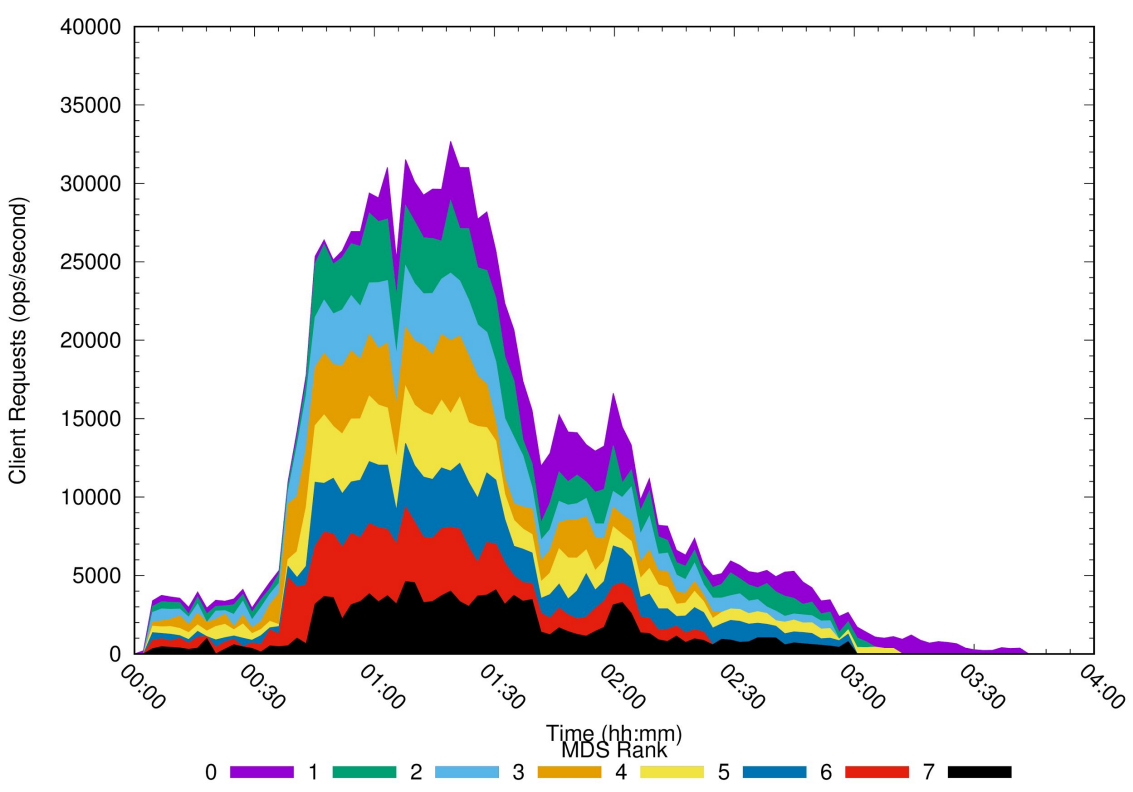
- Each client builds the kernel in an independent sandbox below the root.
- The manual balancer migrates these sandboxes at experiment start across the metadata server cluster, evenly balancing load

```
T=$(mktemp -d tmp.XXXXXX)

(
cd "$T"
wget -q http://download.ceph.com/qa/linux-4.0.5.tar.xz

tar Jxvf linux*.xz
cd linux*
make defconfig
make -j`grep -c processor /proc/cpuinfo`
)

rm -rfv "$T"
```

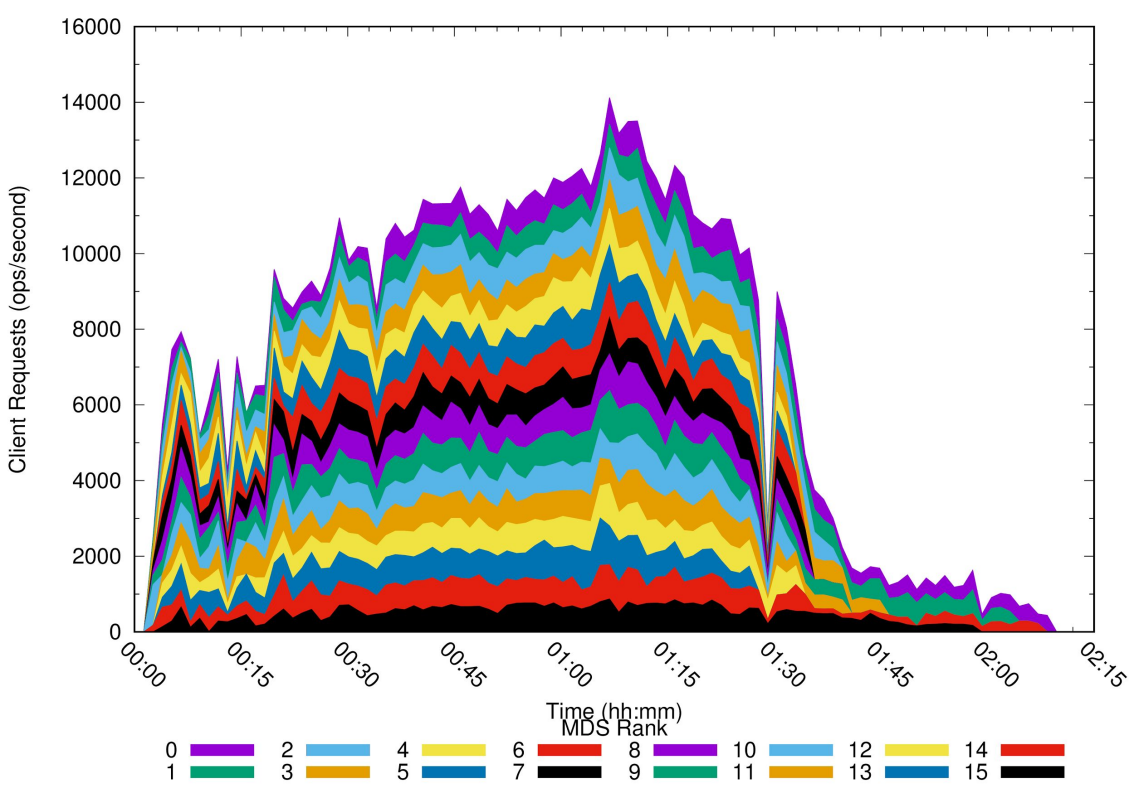
4 MDS and 64 clients



8 MDS and 64 clients



16 MDS and 64 clients



Issues Found

- “mds: stray count remains static after workflows complete”
  http://tracker.ceph.com/issues/19239
- “mds: troubling op throughput scaling from 8 to 16 MDS in kernel build test”
  http://tracker.ceph.com/issues/19240
- “mds: stalled clients apparently due to stale sessions”
  http://tracker.ceph.com/issues/18641
- “osd: unclear error when authentication with monitors fails”
  http://tracker.ceph.com/issues/18629
- “multimds: assertion failure during directory migration”
  http://tracker.ceph.com/issues/17606
- “multimds: MDSs make uneven progress on independent workloads”
  http://tracker.ceph.com/issues/19243

Thanks!

Patrick Donnelly

pdonnell@redhat.com

Thanks to my team: John Spray, Greg Farnum, Zheng Yan, Ramana Raja, Doug Fuller, Jeff Layton, and Brett Niver.

Thanks to Danielle Womboldt, Bryan Stillwell, Sumanta Banerji, Ian Colle for arranging this meeting!