

Revisiting CephFS MDS and mClock QoS Scheduler



Ceph Korea
Community Seminar
2021 04 14



LINE Cloud Storage
Yongseok Oh

Outline of Contents

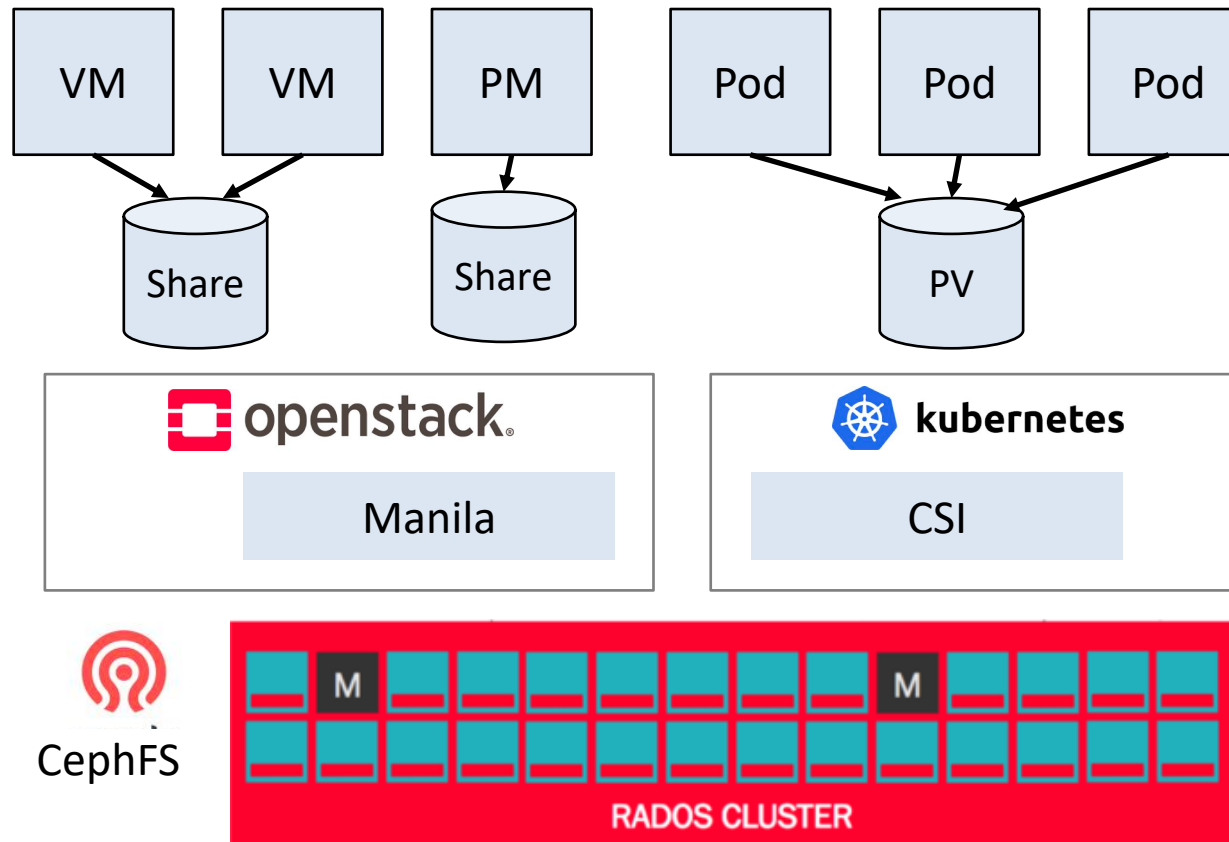
- CephFS MDS Overview
- MDS Evaluation
 - MDS Scalability
 - CephFS Kernel vs FUSE clients
 - Impact of MDS Cache Size
 - Static Subtree Pinning
 - MDS Recovery Time
- mClock QoS Scheduler for CephFS
- Summary

Key Features of CephFS

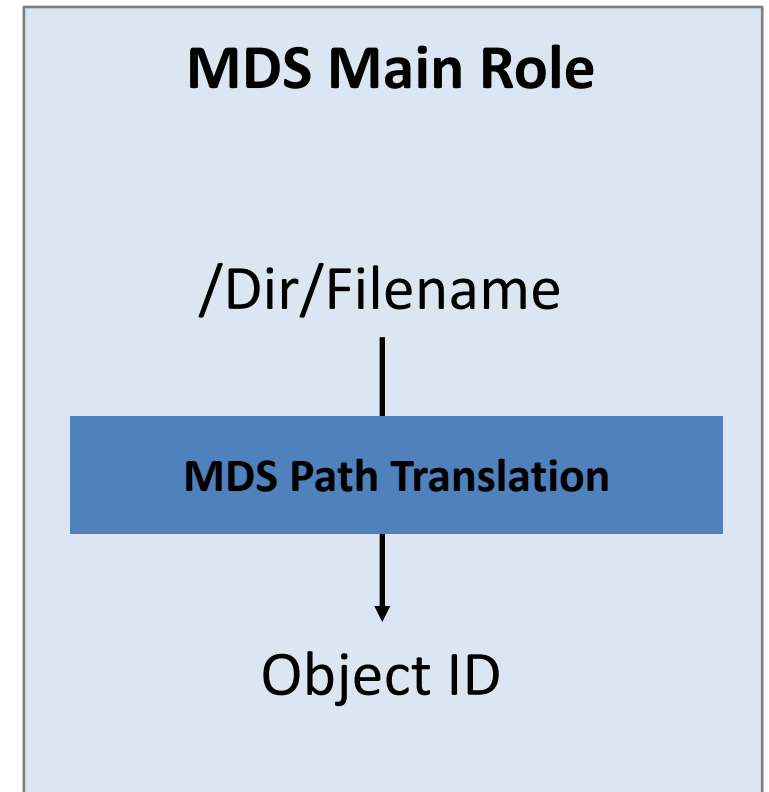
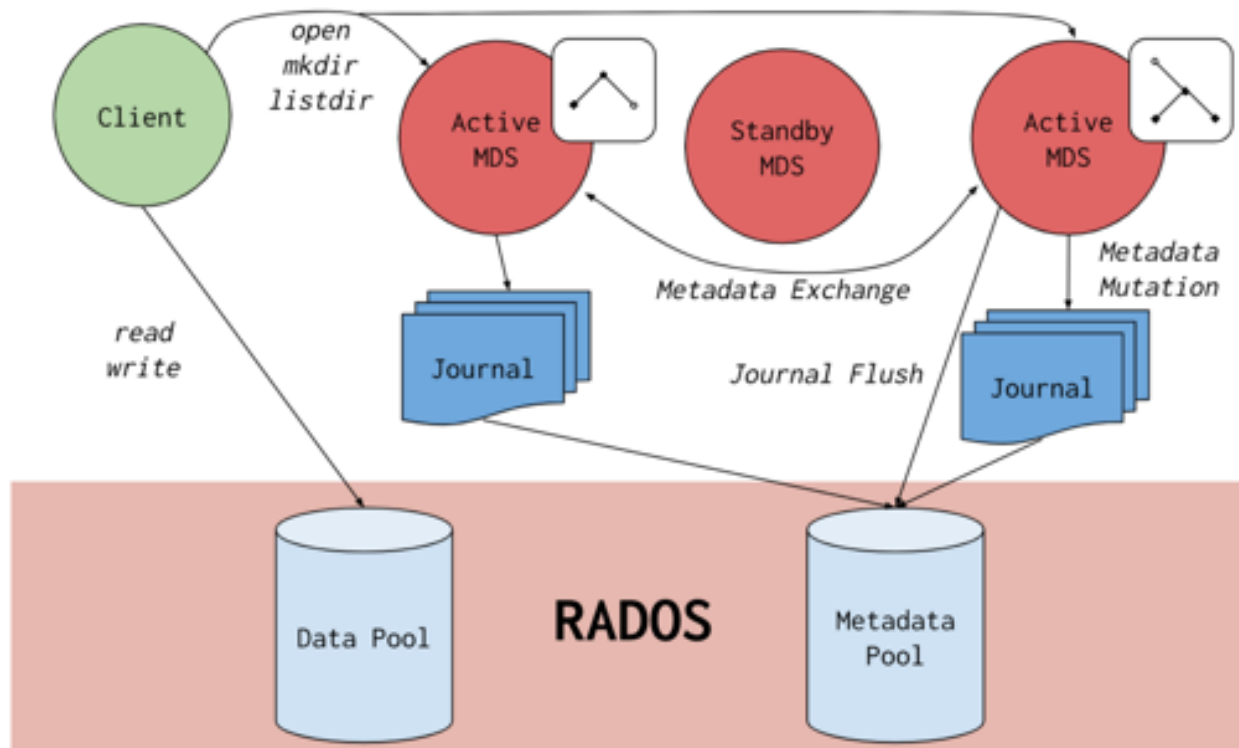
- POSIX compliant distributed file system
- Multi active MDSs (e.g., scalability)
- Standby/standby-replay MDSs (e.g., rapid failover)
- Journaling (e.g., guaranteeing metadata consistency)
- Dynamic/static sub directory partitioning
- Kernel/FUSE/libcephfs client support
- Subvolume/snapshot/quota management
- QoS (not support)

Use Case of CephFS

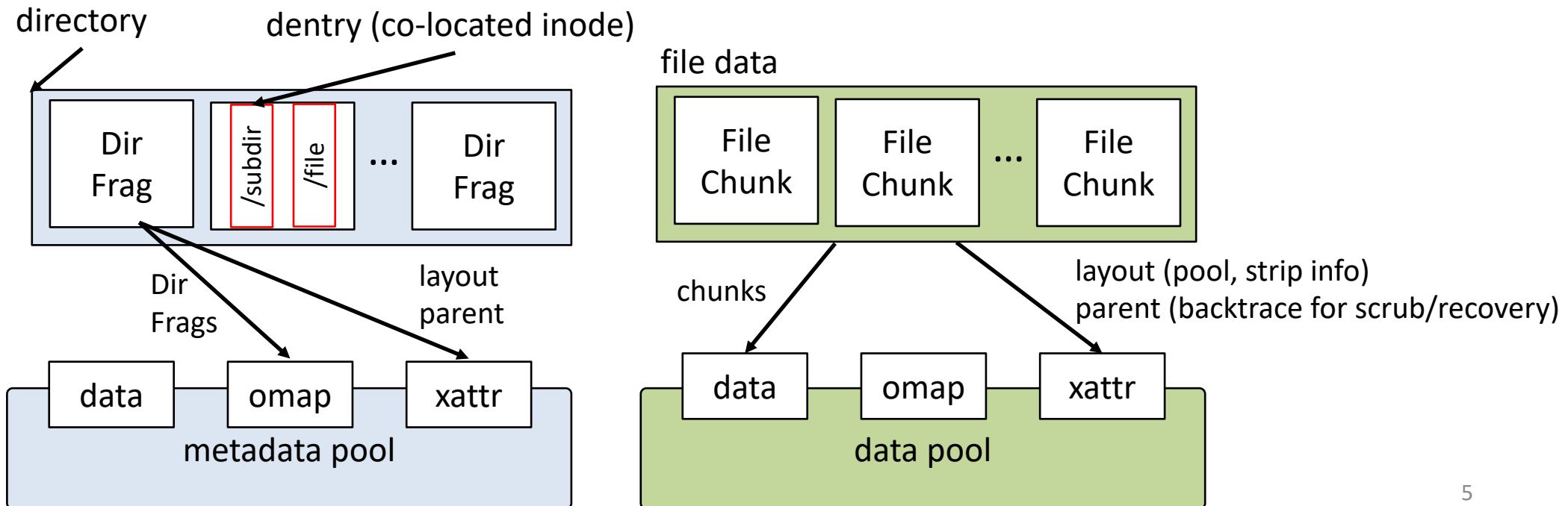
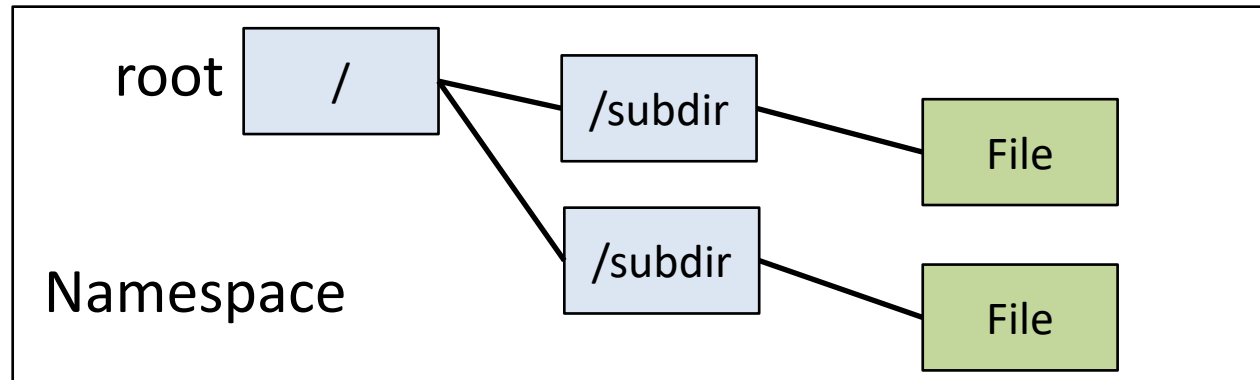
- NAS, ML training, speech data, backup
 - File system can be shared to VMs/Pods (major benefit)



CephFS Architecture



MDS Data Structures



MDS Object Types

Type	Obj Name	MDS Ino	Description	Omap	Data
root	1.000000000	0x1	Root directory	V	
MDSDIR	100.000000000	0x100 ~	Stray directory ~mds0	V	
Log	200.000000000	0x200 ~	Log event		V
Log Backup	300.000000000	0x300 ~	Log event backup		V
Log Pointer	400.000000000	0x400 ~	Log pointer		V
Purge Queue	500.000000000	0x500 ~	Purge queue journal		V
Stray	600.000000000	0x600 ~	Stray inode ~/mds0/stray0~9	V	
mds_openfiles	mds0_openfiles.0	N/A	Open file table	V	
mds_sessionmap	mds0_sessionmap	N/A	Client session map	V	
mds_inotable	mds0_inotable	N/A	Inode number allocation table		V
User	10000000001.000000000	0x100000000001~0x1 FFFFFFFF	User files/directories	V	V

List OMAP Keys

- dir/file names are stored as keys in OMAP

```
> cd /mnt

> mkdir dir_{1..10}

> ls
dir_1  dir_10  dir_2  dir_3  dir_4  dir_5  dir_6  dir_7  dir_8  dir_9

> bin/rados -c ceph.conf -k keyring -p cephfs.a.meta listomapkeys 1.00000000
dir_1_head
dir_2_head
dir_3_head
dir_4_head
dir_5_head
dir_6_head
dir_7_head
dir_8_head
dir_9_head
dir_10_head
```


List OMAP Values

- inode contents are stored as values in OMAP

```
> bin/rados -c ceph.conf -k keyring -p cephfs.a.meta listomapvals 1.00000000
```

```
dir_1_head
```

```
value (462 bytes) :
```

00000000	02 00 00 00 00 00 00 00	49 0f 06 a3 01 00 00 00I.....
00000010	00 00 00 00 01 00 00 00	00 00 00 30 39 75 60 fb09u`.
00000020	81 fb 22 ed 41 00 00 f8	2a 00 00 f8 2a 00 00 01	..".A...*...*...
00000030	00 00 00 00 02 00 00 00	00 00 00 00 02 02 18 00
00000040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 ff ff
00000050	ff ff ff ff ff ff 00 00	00 00 00 00 00 00 00 00
00000060	00 00 01 00 00 00 ff ff	ff ff ff ff ff ff 00 00
00000070	00 00 00 00 00 00 00 00	00 00 30 39 75 60 fb 8109u`.
00000080	fb 22 30 39 75 60 fb 81	fb 22 00 00 00 00 00 00	."09u`...".....
00000090	00 00 03 02 28 00 00 00	00 00 00 00 00 00 00 00(.....
000000a0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

```
... (omitting)
```

Outline of Contents

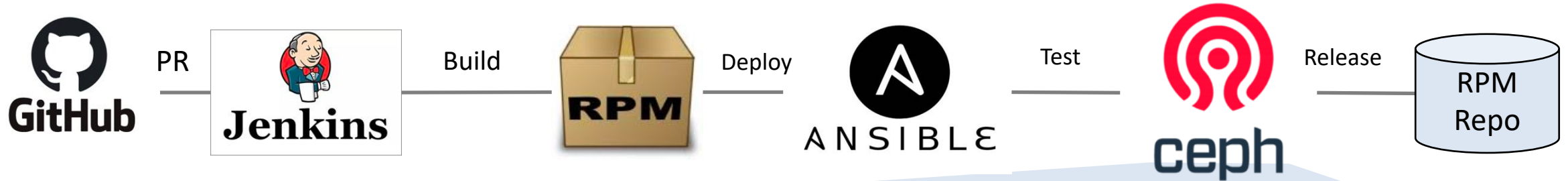
- CephFS MDS Overview
- **MDS Evaluation**
 - MDS Scalability
 - CephFS Kernel vs FUSE clients
 - Impact of MDS Cache Size
 - Static Subtree Pinning
 - MDS Recovery Time
- mClock QoS Scheduler for CephFS
- Summary & Wrap Up

Evaluation Configuration

- Ceph Nautilus 14.2.16
- CentOS 7.9 (kernel 3.10.0-1160.11.1.el7.x86_64)
- Benchmark
 - smallfile, vdbench, kernel compile
- Server spec.

Type	Spec	Count
MDS	4 * vCPU 32GB RAM	16
OSD	40 * pCPU 128GB RAM	9 Servers * 6 SATA SSDs = 54
Mon	2 * vCPU 4GB RAM	3
Mgr	4 * vCPU 8GB RAM	3
Client	4 * vCPU 8GB RAM	20
Network	10Gbps	

Ceph CI/CD System in LINE



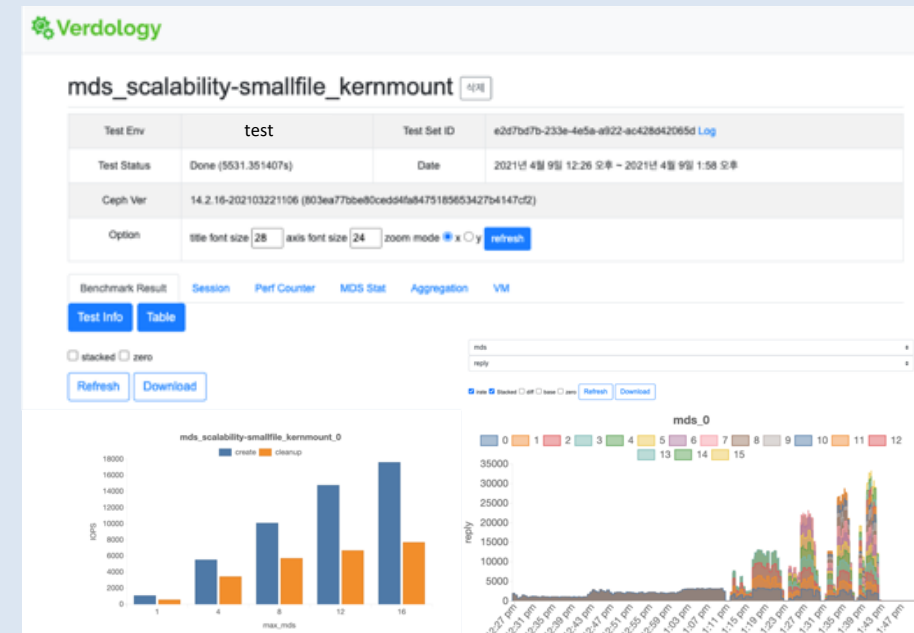
Ceph Test Frameworks

[teuthology]

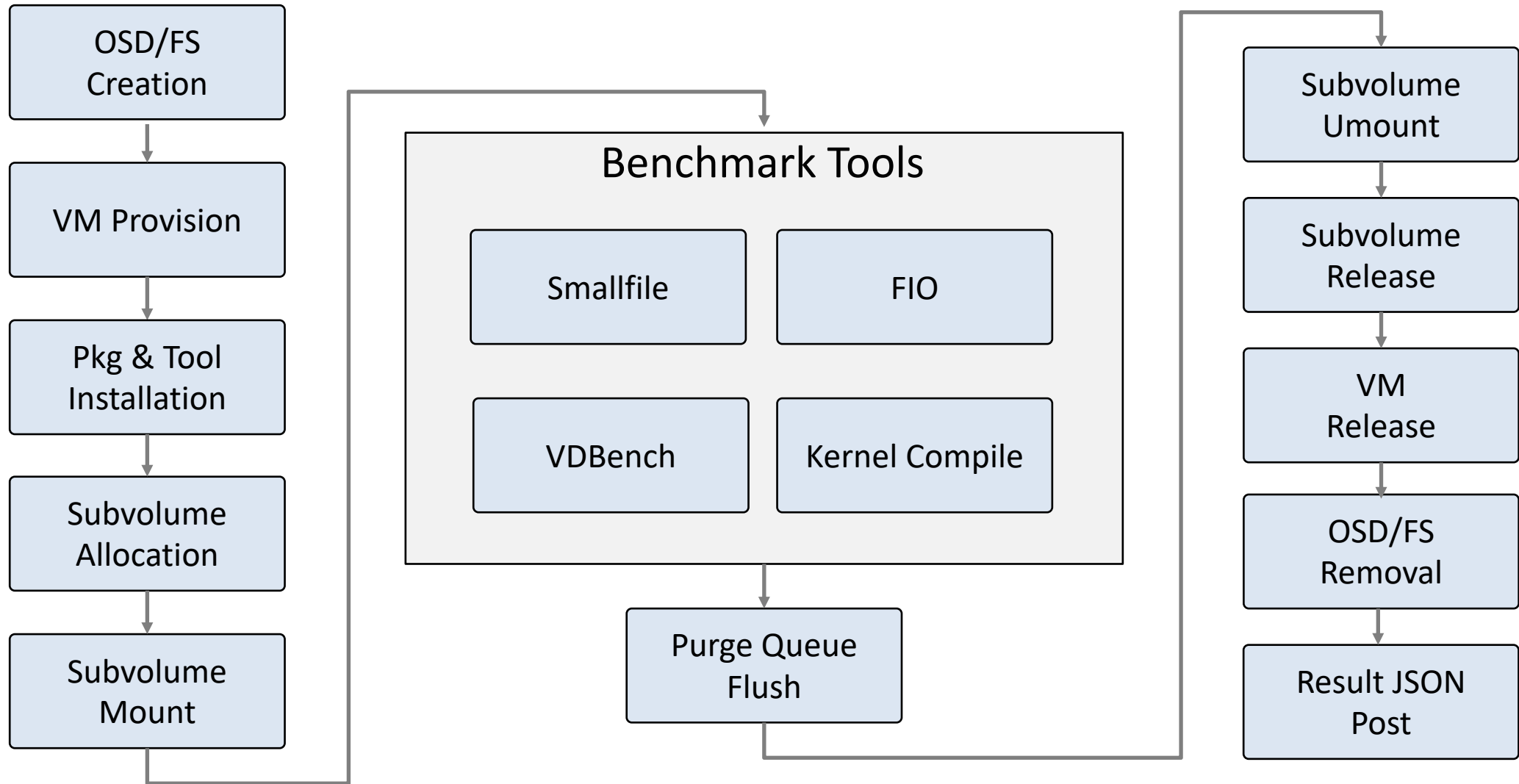
[verdology]

The screenshot shows the Teuthology web interface. At the top, there's a navigation bar with 'Pulpito' and various filters like 'Queue', 'By Status', 'By Branch', 'By Suite', 'By Machine Type', and a search bar. Below this, a summary bar shows the status of the test runs: 2 Queued, 3 Failed, 1 Running, 1 Waiting, 28 Passed, and 35 Total. The main table lists individual test runs with columns for Status, Job ID, Links, Posted, Started, Updated, Runtime, Duration, In Waiting, Machine, Teuthology Branch, OS Type, OS Version, and Nodes. The table shows a list of 20 test runs, all of which passed.

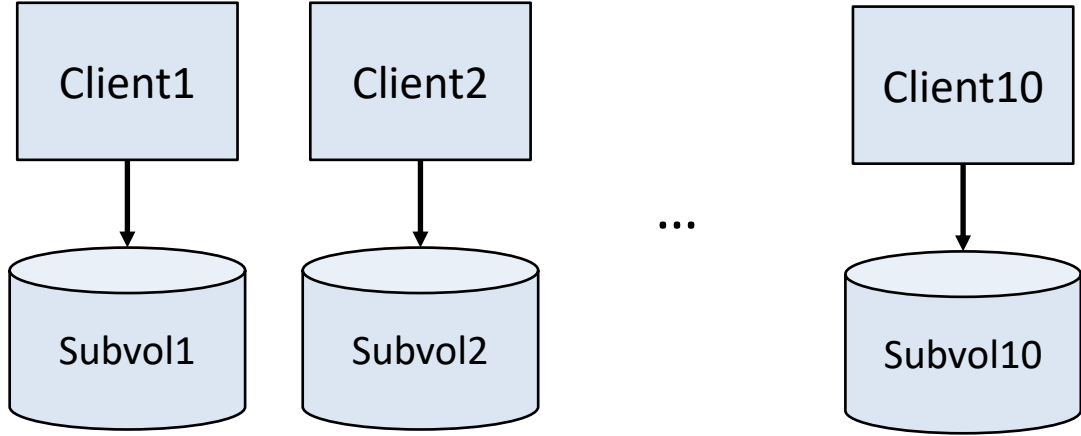
Status	Job ID	Links	Posted	Started	Updated	Runtime	Duration	In Waiting	Machine	Teuthology Branch	OS Type	OS Version	Nodes
pass	187		2021-04-09 01:41:22	2021-04-09 03:12:41	2021-04-09 03:40:40	0:27:59	0:25:31	0:02:28	vps	master	centos	7.8	2
pass	188		2021-04-09 01:41:23	2021-04-09 03:12:43	2021-04-09 03:34:42	0:21:59	0:18:42	0:03:17	vps	master	centos	7.8	2
pass	189		2021-04-09 01:41:24	2021-04-09 03:34:44	2021-04-09 04:08:43	0:33:59	0:31:56	0:02:03	vps	master	centos	7.8	2
pass	190		2021-04-09 01:41:25	2021-04-09 03:40:42	2021-04-09 04:02:41	0:21:59	0:18:34	0:03:25	vps	master	centos	7.8	2
pass	191		2021-04-09 01:41:26	2021-04-09 04:02:43	2021-04-09 04:22:42	0:19:59	0:17:21	0:02:38	vps	master	centos	7.8	2
pass	192		2021-04-09 01:41:27	2021-04-09 04:08:45	2021-04-09 04:28:44	0:19:59	0:17:27	0:02:32	vps	master	centos	7.8	2
pass	193		2021-04-09 01:41:28	2021-04-09 04:22:44	2021-04-09 04:54:43	0:31:59	0:28:59	0:03:00	vps	master	centos	7.8	2
pass	194		2021-04-09 01:41:29	2021-04-09 04:28:46	2021-04-09 05:00:45	0:31:59	0:29:42	0:02:17	vps	master	centos	7.8	2
pass	195		2021-04-09 01:41:30	2021-04-09 04:54:45	2021-04-09 05:32:44	0:37:59	0:34:57	0:03:02	vps	master	centos	7.8	2
pass	196		2021-04-09 01:41:31	2021-04-09 05:00:47	2021-04-09 05:22:47	0:22:00	0:19:46	0:02:14	vps	master	centos	7.8	2
pass	197		2021-04-09 01:41:32	2021-04-09 05:22:49	2021-04-09 05:50:48	0:27:59	0:25:25	0:02:34	vps	master	centos	7.8	2
pass	198		2021-04-09 01:41:32	2021-04-09 05:32:46	2021-04-09 05:54:46	0:22:00	0:19:06	0:02:54	vps	master	centos	7.8	2
pass	199		2021-04-09 01:41:33	2021-04-09 05:50:50	2021-04-09 06:12:50	0:22:00	0:18:28	0:03:32	vps	master	centos	7.8	2
pass	200		2021-04-09 01:41:34	2021-04-09 05:54:48	2021-04-09 06:24:47	0:29:59	0:27:50	0:02:09	vps	master	centos	7.8	2



CephFS Test Flow of Verdology



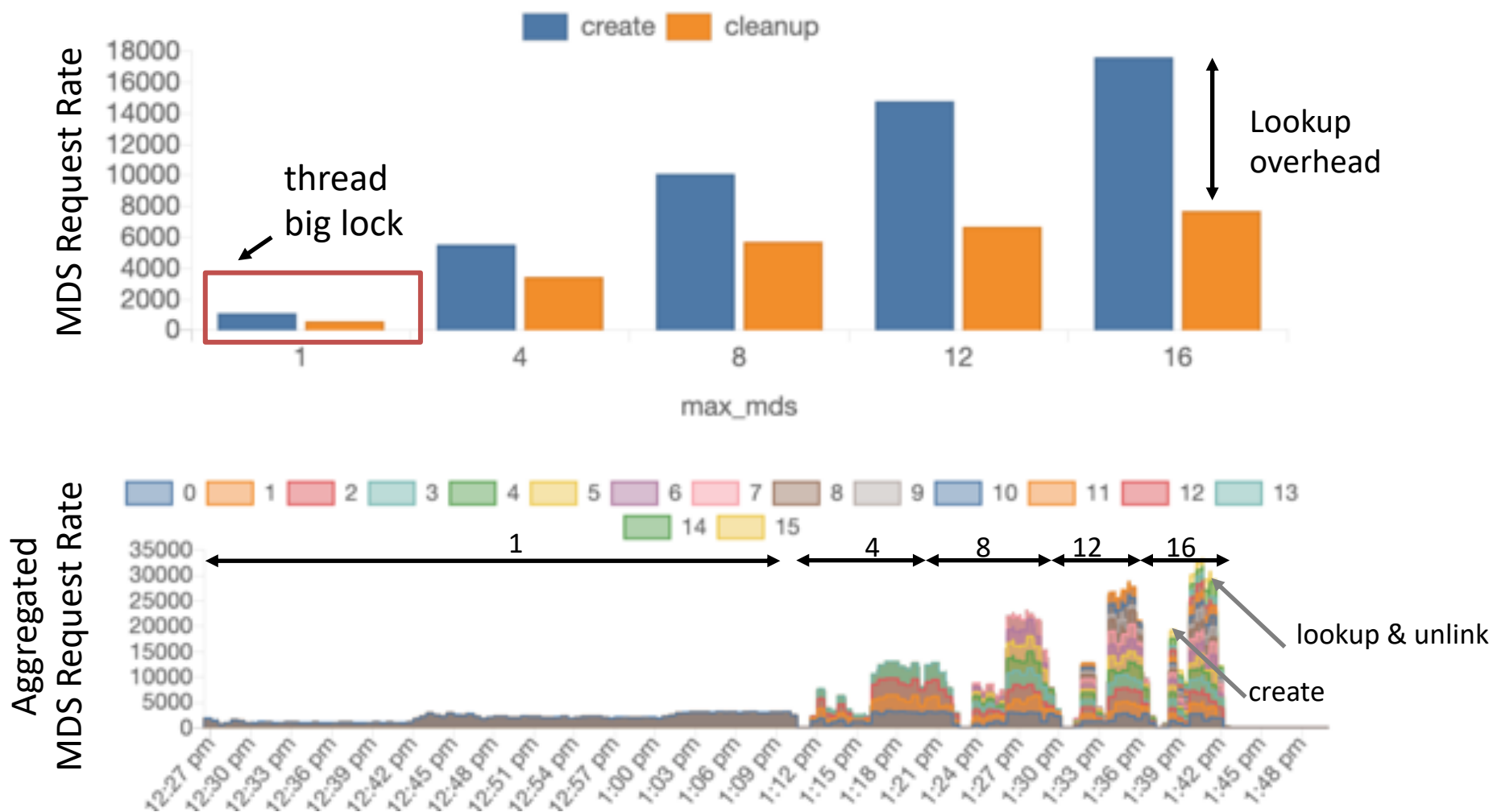
Benchmark Configuration for MDS Scalability



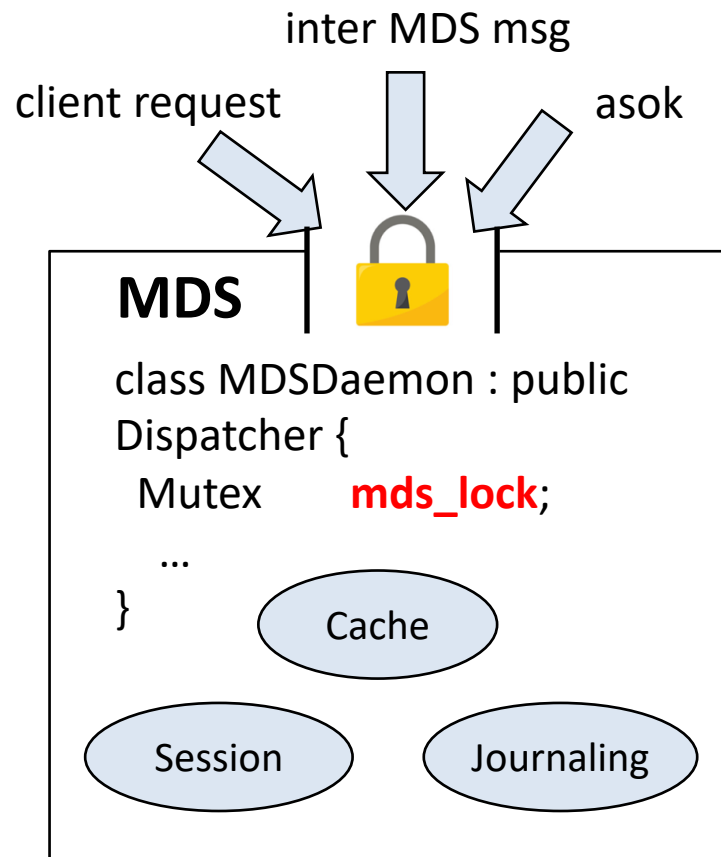
Each client is working on its own subvolume.

Type	Spec	Remark
Tool	Smallfile	
max_mds	1~16	
# of Clients	10	
Files per Client	50,000	
File Size	0	
Files per Dir	1000	
Operation Type	Create, Unlink	
Thread	1	
CephFS Client	Kernel cephfs driver	

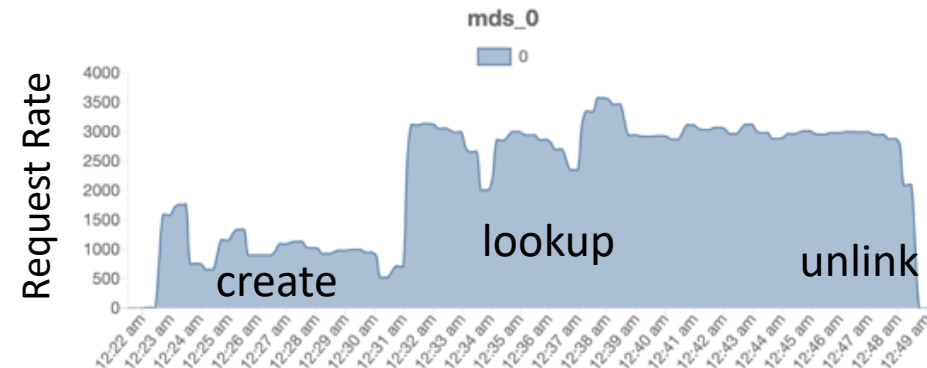
MDS Scalability Results



Less Scalable with Big Lock



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
30589	ceph	20	0	13.1g	12.7g	30780	R	99.0	54.1	86:32.68	ms_dispatch
30580	ceph	20	0	13.1g	12.7g	30780	S	21.8	54.1	32:58.47	msgr-worker-2
52410	ceph	20	0	13.1g	12.7g	30780	S	19.8	54.1	25:14.88	md_submit
30579	ceph	20	0	13.1g	12.7g	30780	R	18.8	54.1	18:39.49	msgr-worker-1
30578	ceph	20	0	13.1g	12.7g	30780	R	13.9	54.1	22:56.11	msgr-worker-0
52404	ceph	20	0	13.1g	12.7g	30780	S	4.0	54.1	3:48.57	OpHistorySvc
30585	ceph	20	0	13.1g	12.7g	30780	S	3.0	54.1	3:01.88	log

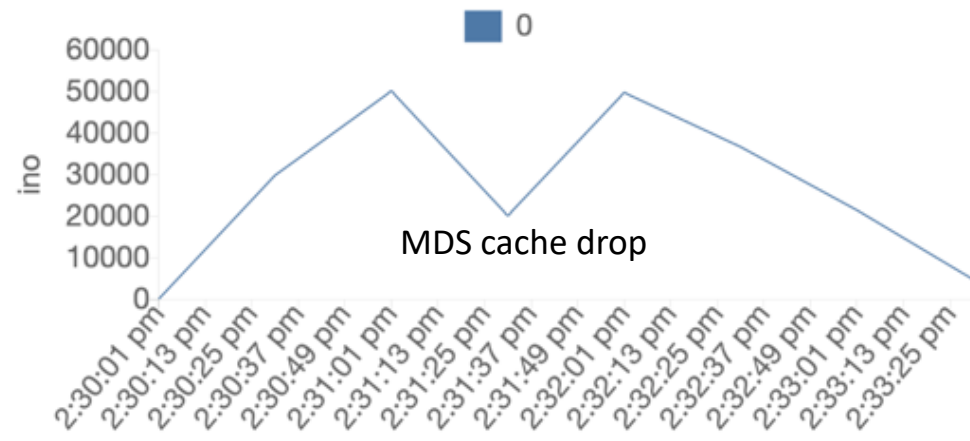
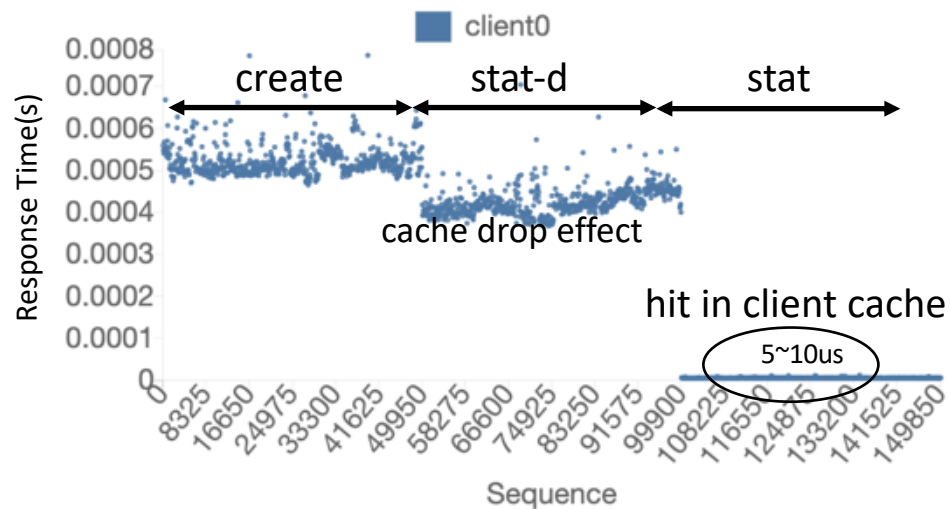
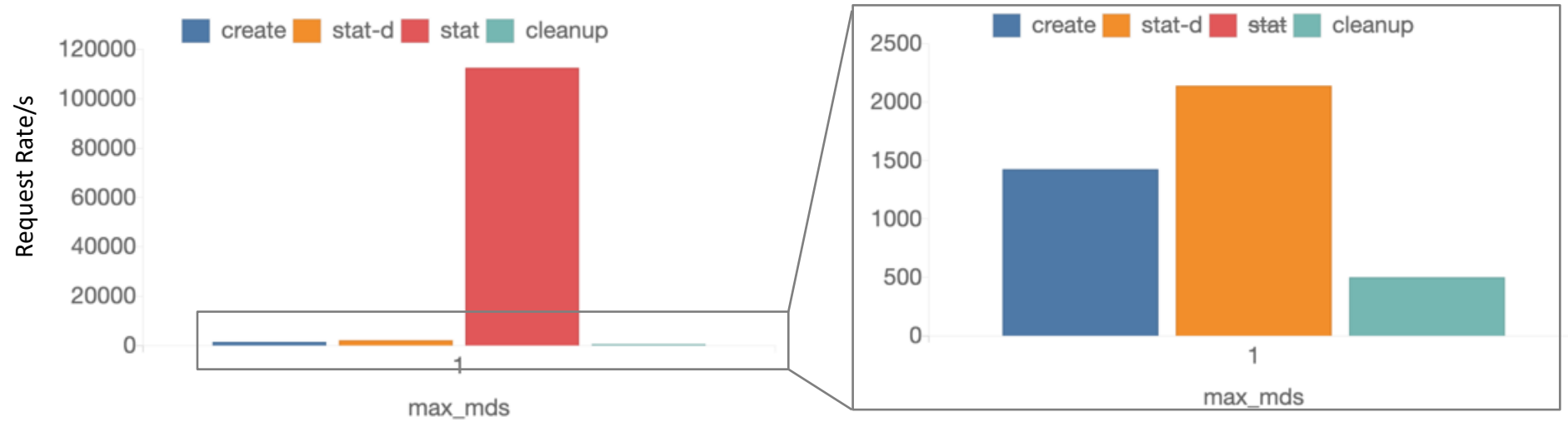


Test Parameters for Comp. of Kernel and FUSE

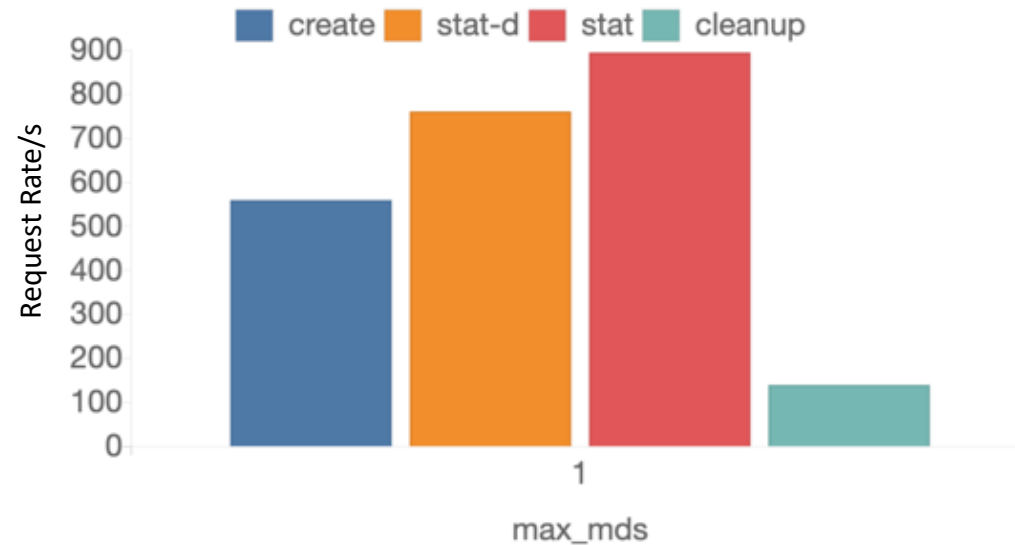
Type	Spec	Remark
# of Active MDSs (max_mds)	1	
# of Clients	1	
CephFS Client	Kernel cephfs driver, ceph-fuse	
Tool	Smallfile	
Files per Client	50,000	
File Size	0	
Files per Dir	1000	
Operation Type	Create, Stat-d, Stat, Unlink	
Thread	1	

- Stat-d (w mds cache drop)
 - Stat operations are performed after mds caches are dropped (e.g., ceph daemon mds.\$(hostname) cache drop)
- Stat (e.g., cache hit case)
 - Metadata contents could be found in either MDS rank or client

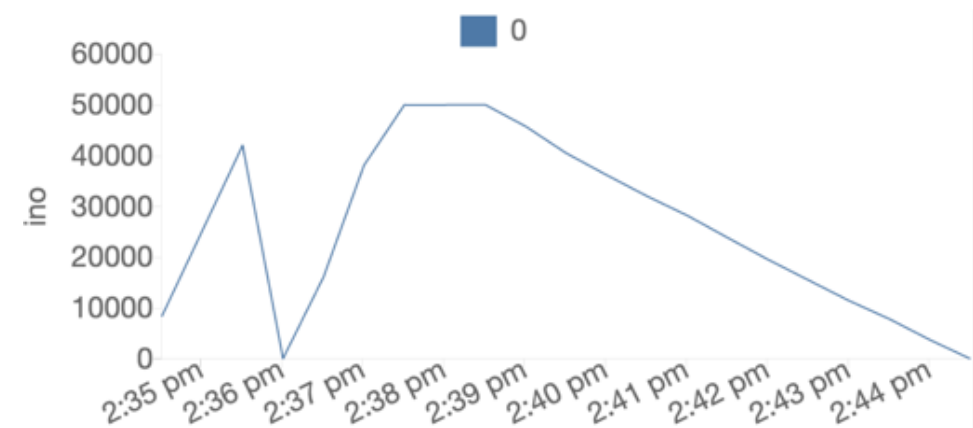
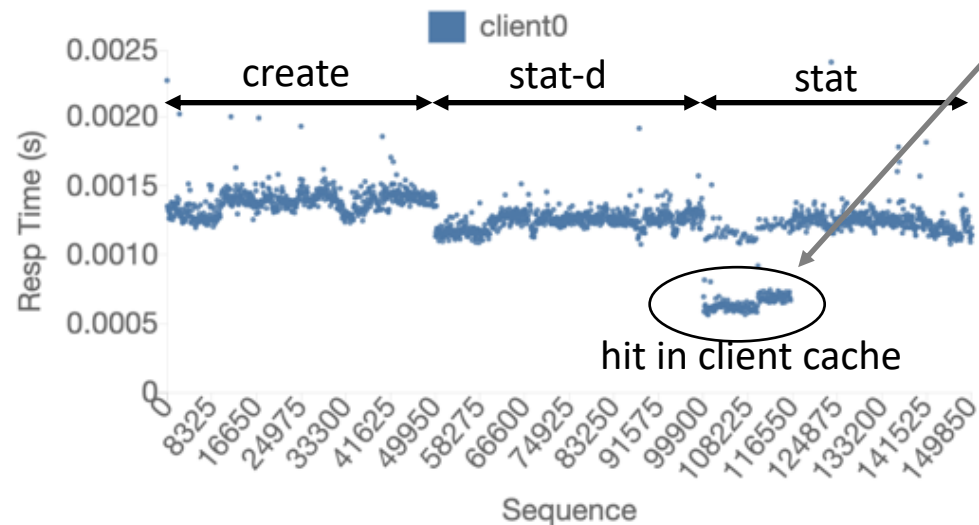
Kernel based Client



FUSE based Client

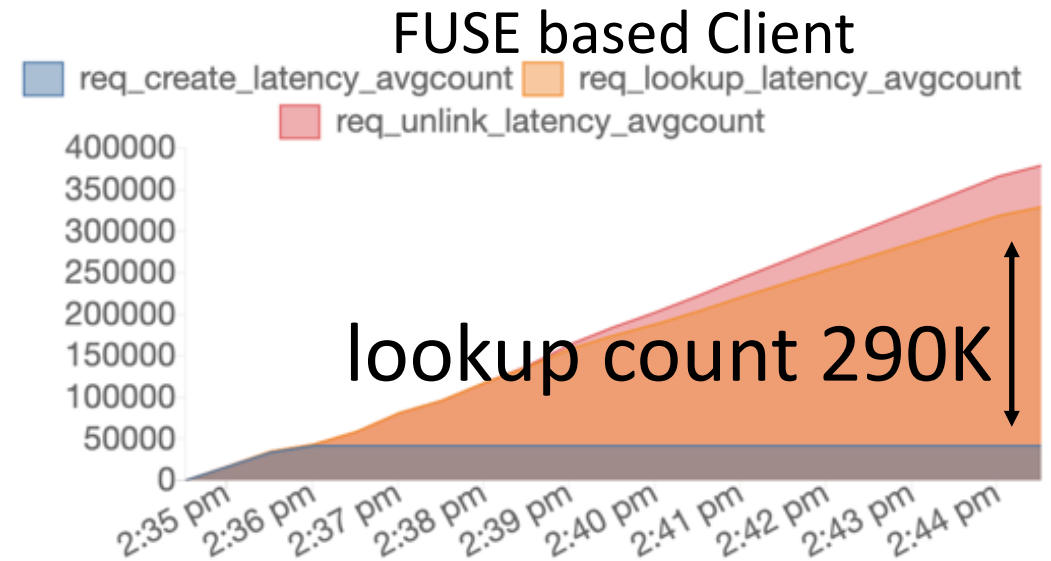
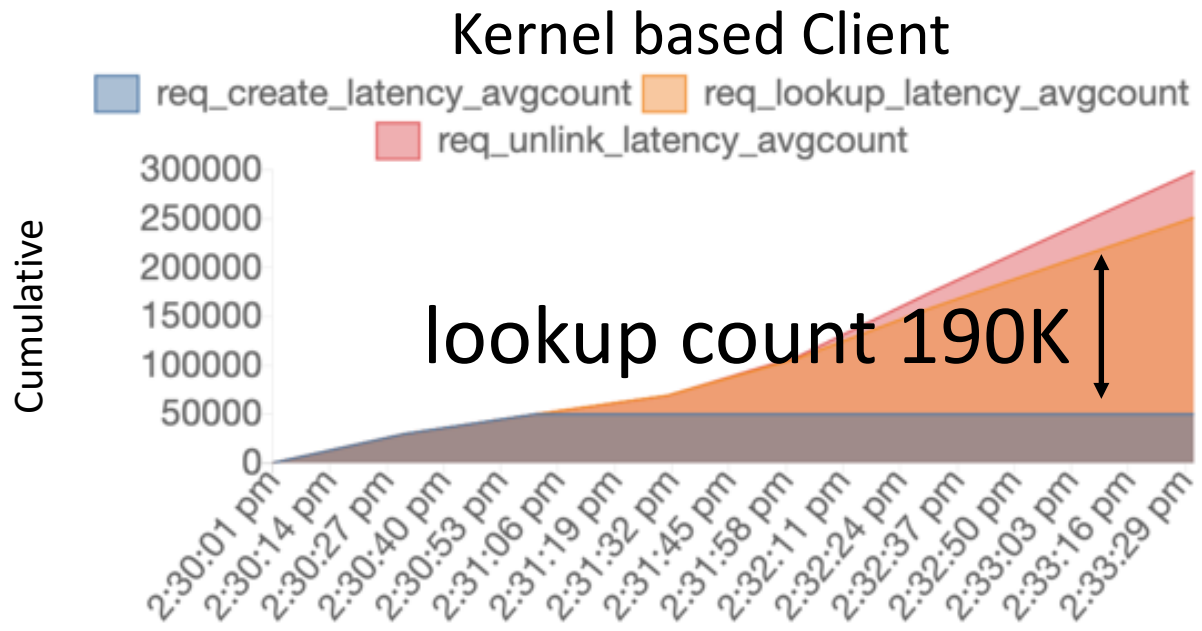


FUSE Client Configuration
client_cache_size = 16384 (by default)



Summary of Results

- Kernel CephFS driver provides superior performance
 - Utilize VFS inode/dentry caches, resulting in reduced lookup count
 - Minimize process context switch and data copy between user-space and kernel space
 - But, kernel driver suffers from some technical issues, rebooting system is required



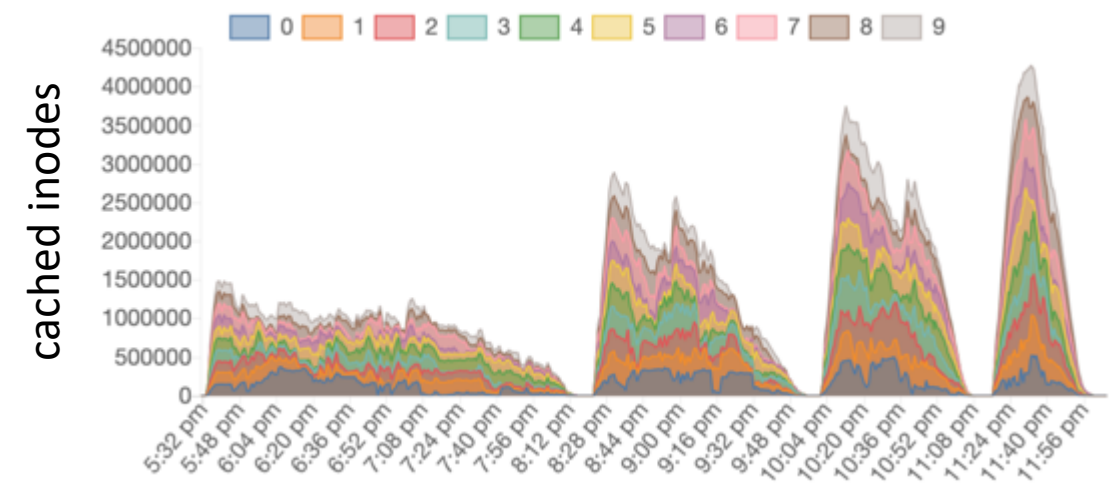
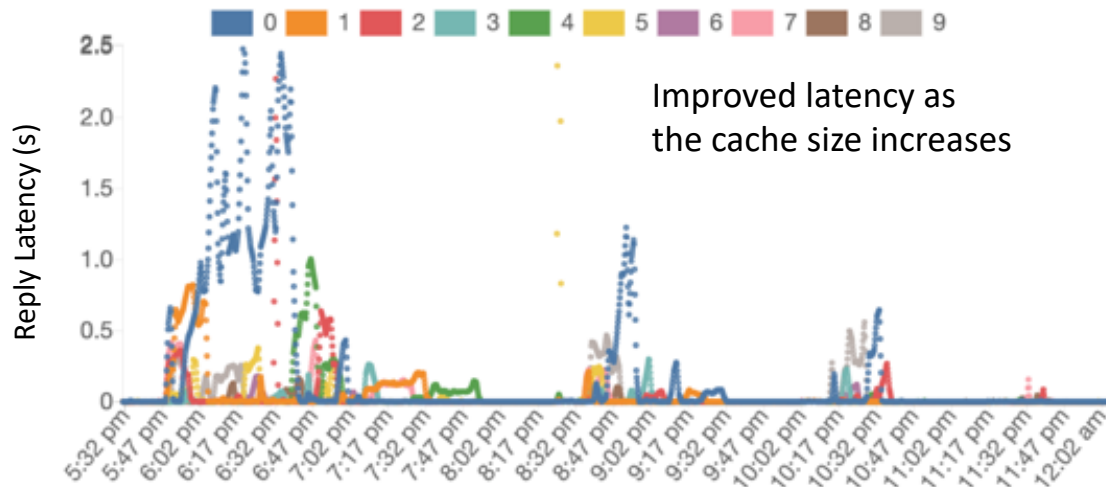
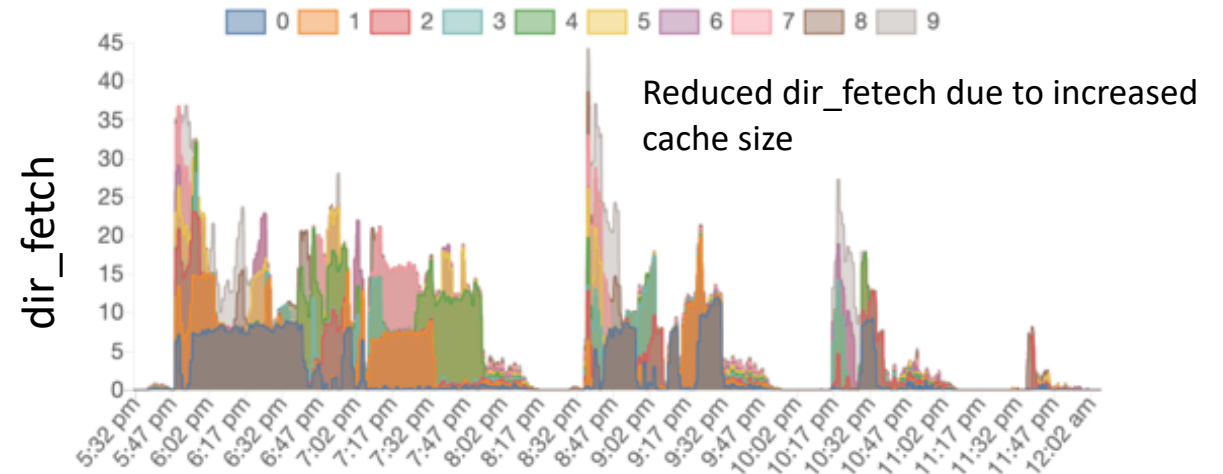
For simplicity and quick experiments, we have utilized kernel CephFS driver unless mentioned otherwise!

Test Parameters for MDS Cache Size

Type	Spec		Remark
# of Active MDSs (max_mds)	10		
Total_cache_size for all MDSs	4GB, 8GB, 12GB, 16GB (Example, if the total cache size is 4GB, each MDS has 4GB/10 MDSs)		
# of Clients	20		
CephFS Client	Kernel cephfs driver		
Tool	Smallfile	Kernel compile	
Files per Client	200K = 4Mill files / 20 clients	20 clients	
File Size	0	-	
Files per Dir	20K	-	
Operation Type	Create, Stat, Unlink	-	
Thread	1	1	

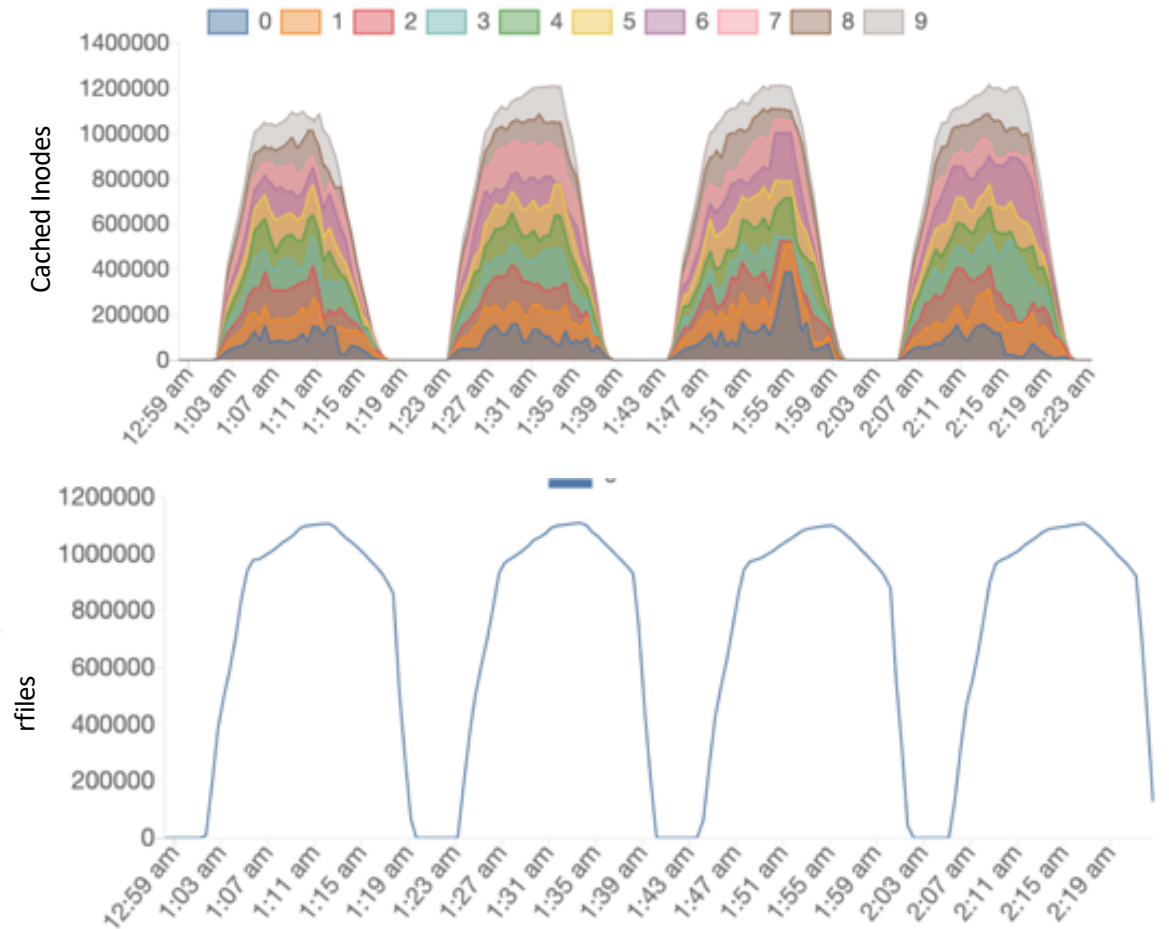
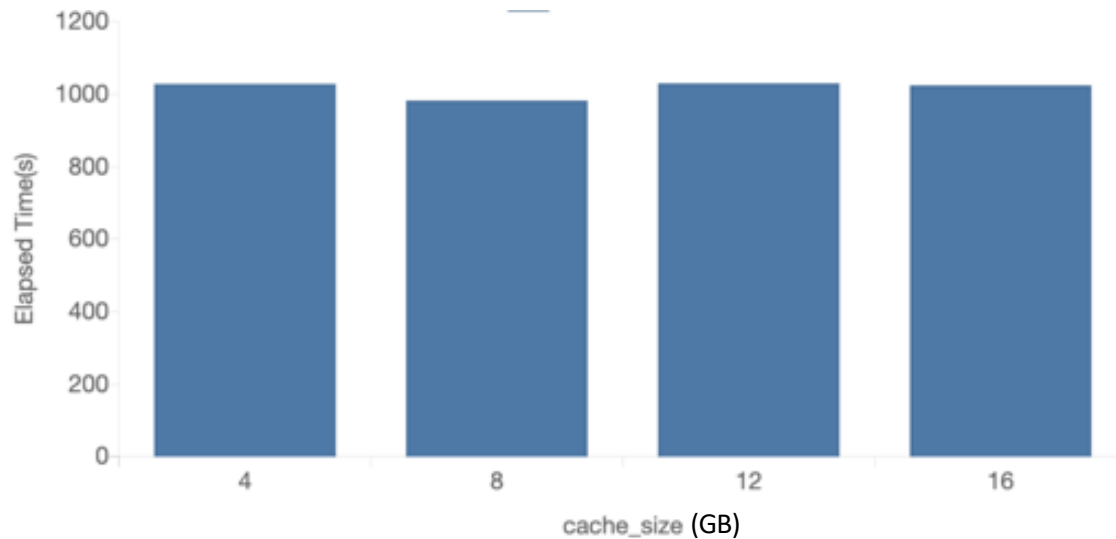
- Assumption
 - 350K inodes (including dentries) per GB (mds_cache_memory_limit)
 - 16GB MDS caches can retain more than 4Mill inodes approximately
 - To analyze impact of cache size, we vary the total MDS size from 4 to 16GB

Impact of MDS Caches (smallfile)



Impact of MDS Caches (kernel compile)

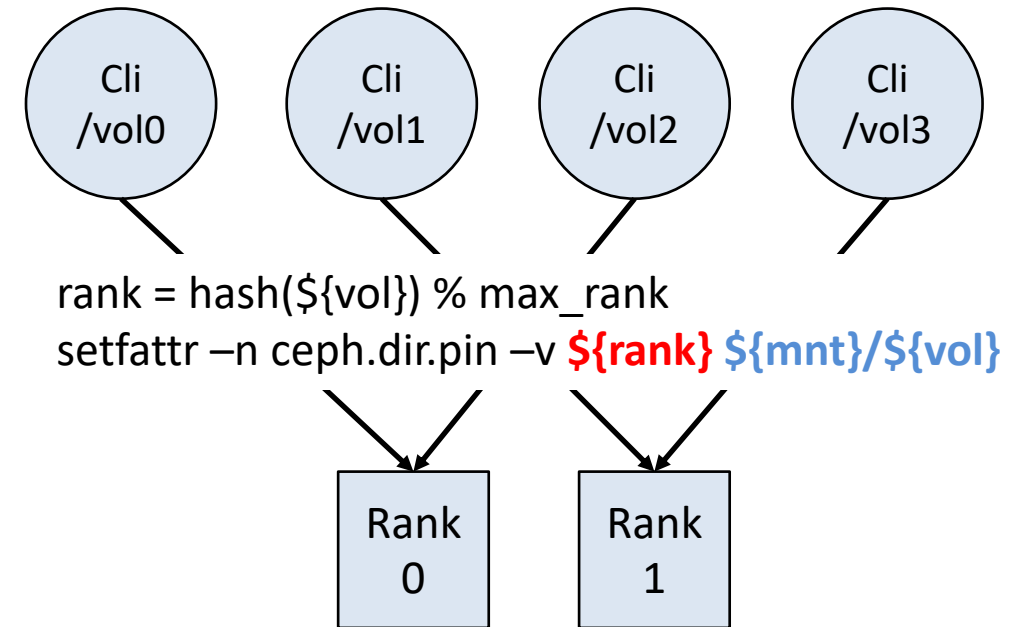
- Caches are enough to keep inodes in RAM



Test Configurations for Subtree Pinning

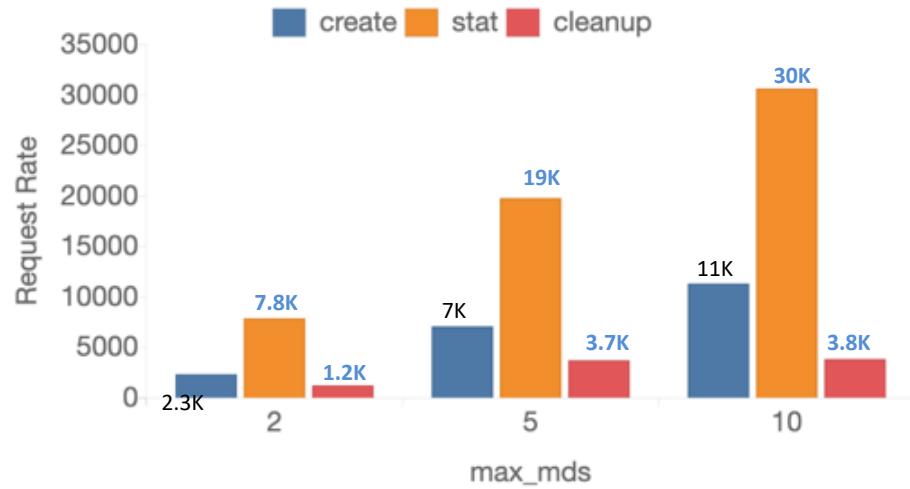
Type	Spec	
# of Active MDSs (max_mds)	2, 5, 10	
Per MDS Cache Size	1GB (mds_cache_memory_limit)	
# of Clients	20	
CephFS Client	Kernel cephfs driver	
Tool	Smallfile	Kernel compile
Files per Client	350K * max_mds / 20 (# of clients)	-
File Size	0	-
Files per Dir	17K (350K / 20)	-
Operation Type	Create, Stat, Unlink	-
Thread	1	1

Example of Subtree Pinning

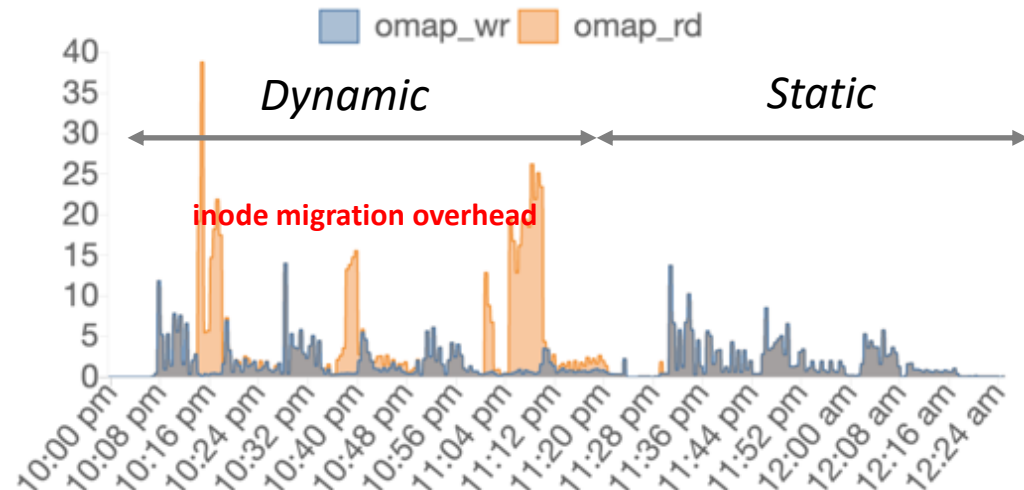
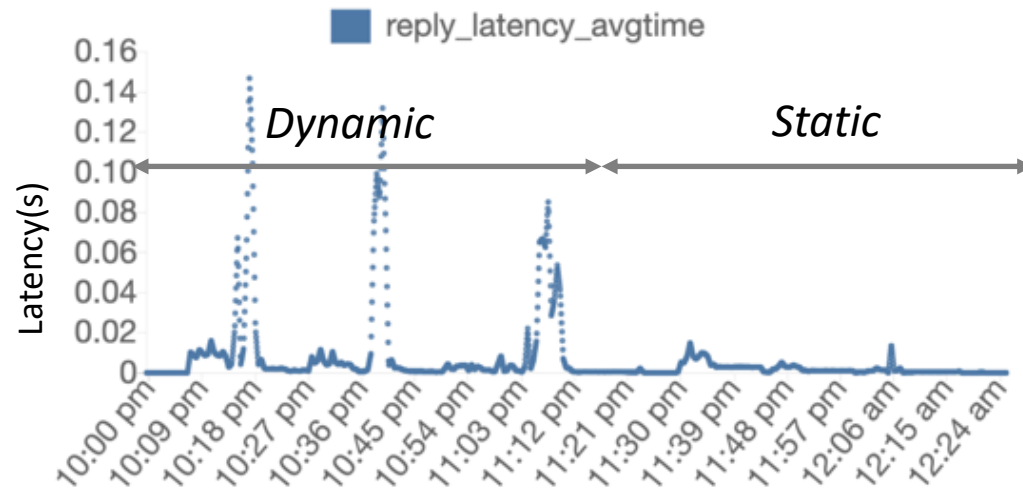
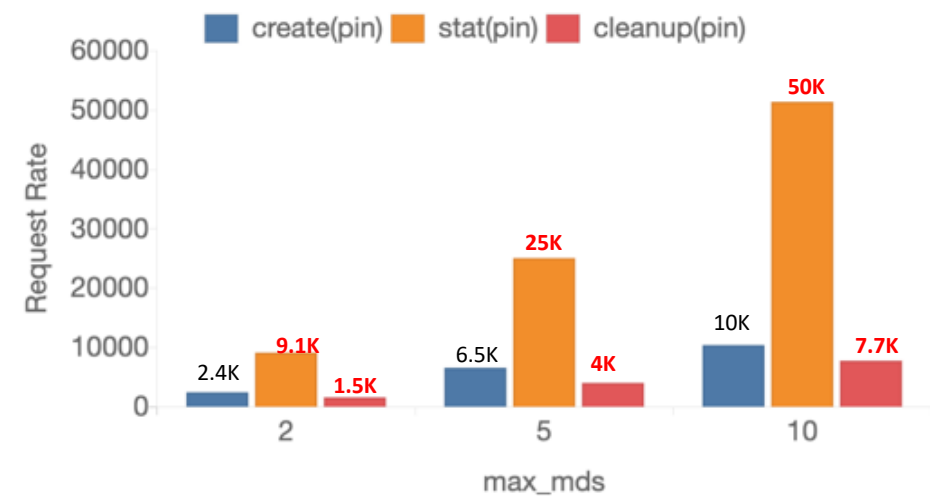


Subtree Pinning

Dynamic Balancing (Default)



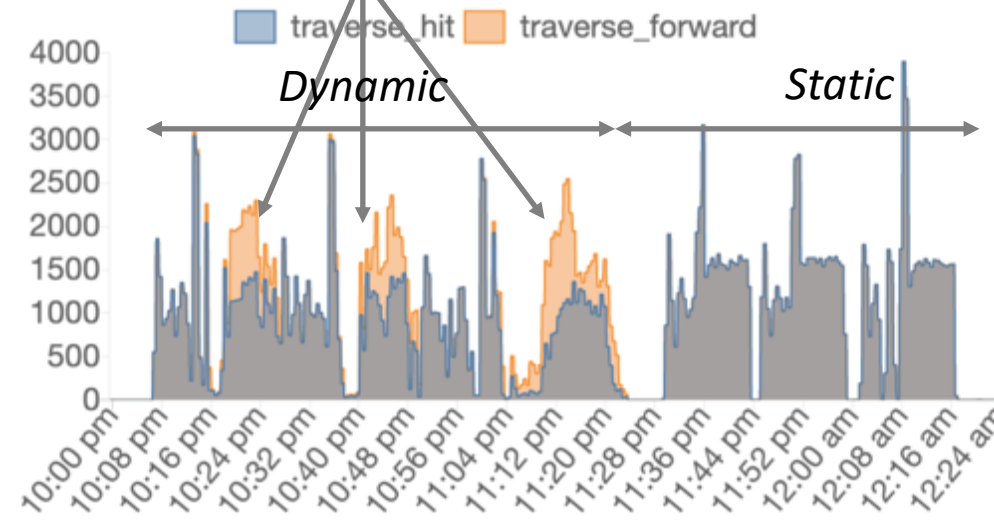
Static Pinning



Dynamic balancing hurts internal lock and metadata update costs, leading to increased latencies

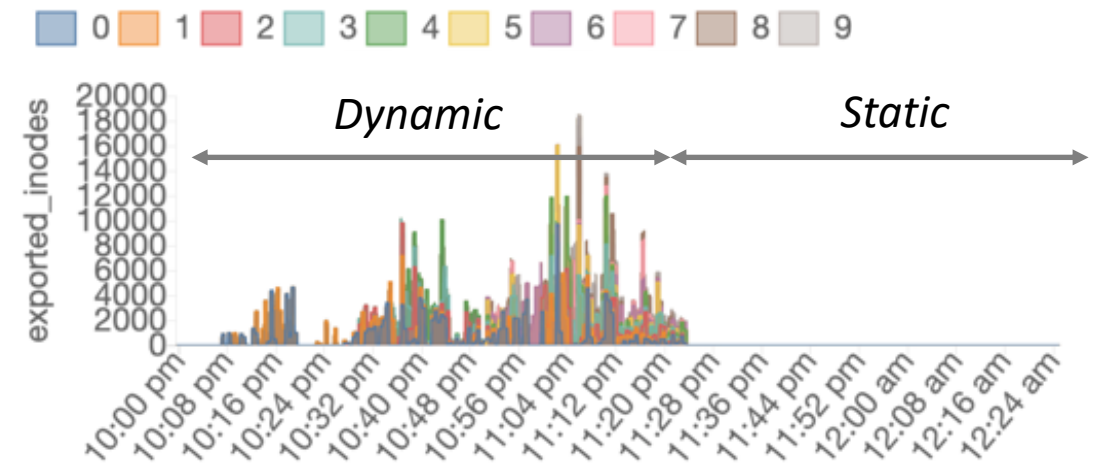
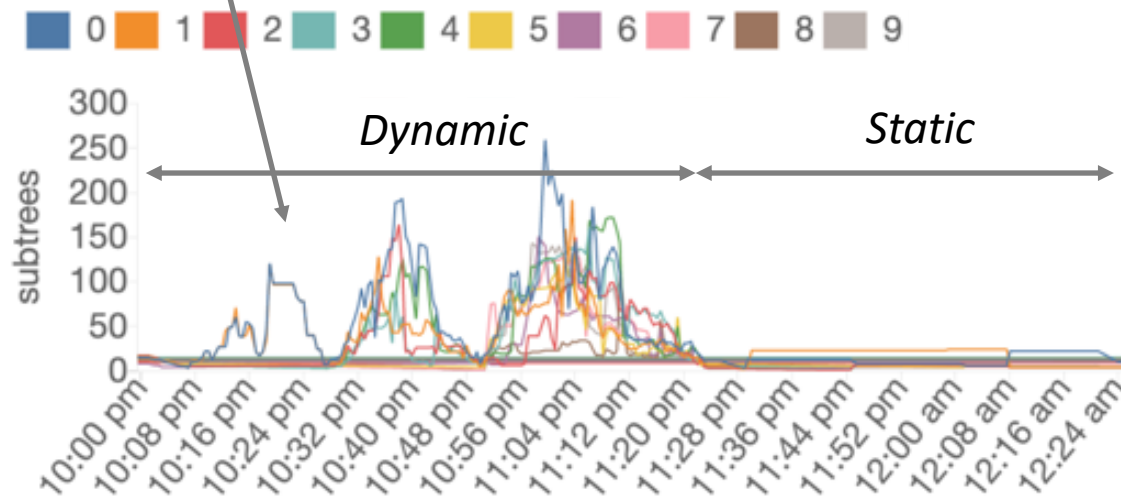
Subtree Pinning

Forward traverse request to other ranks

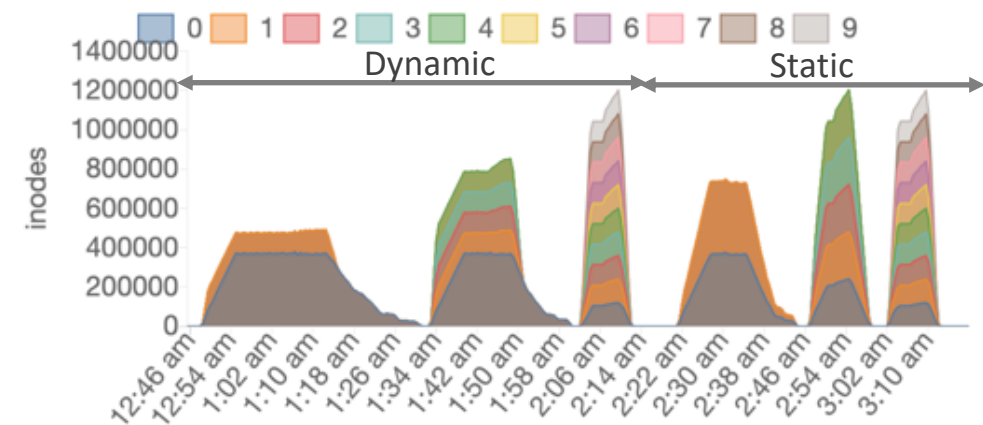
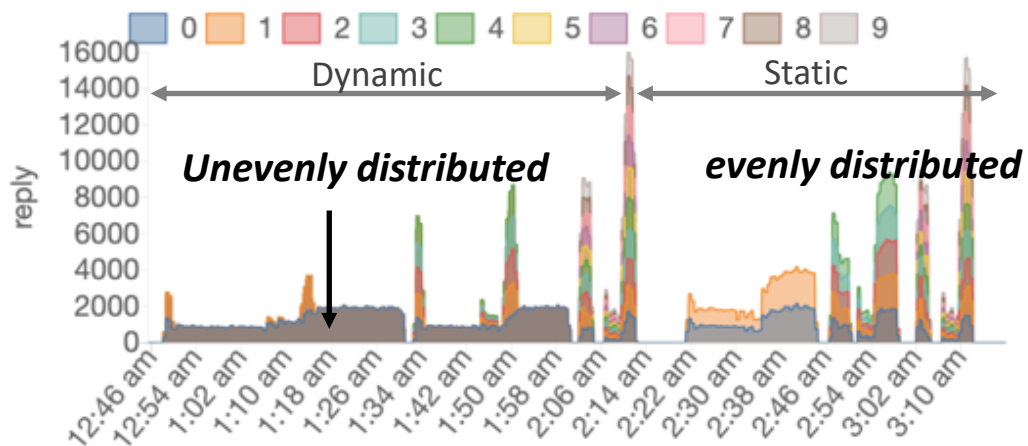
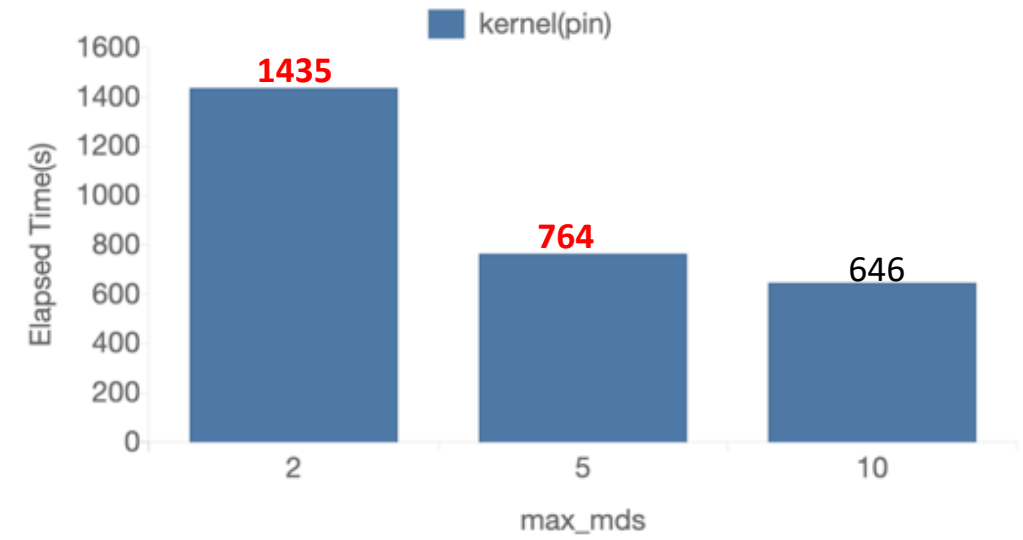
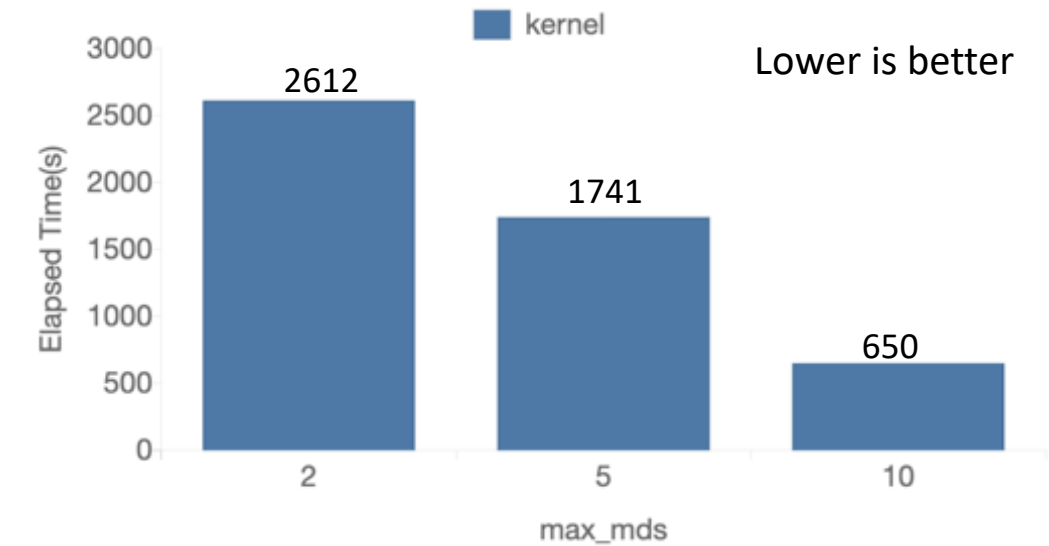


Rank0

Increased metadata management cost (e.g., journaling)



Subtree Pinning – Kernel Compile

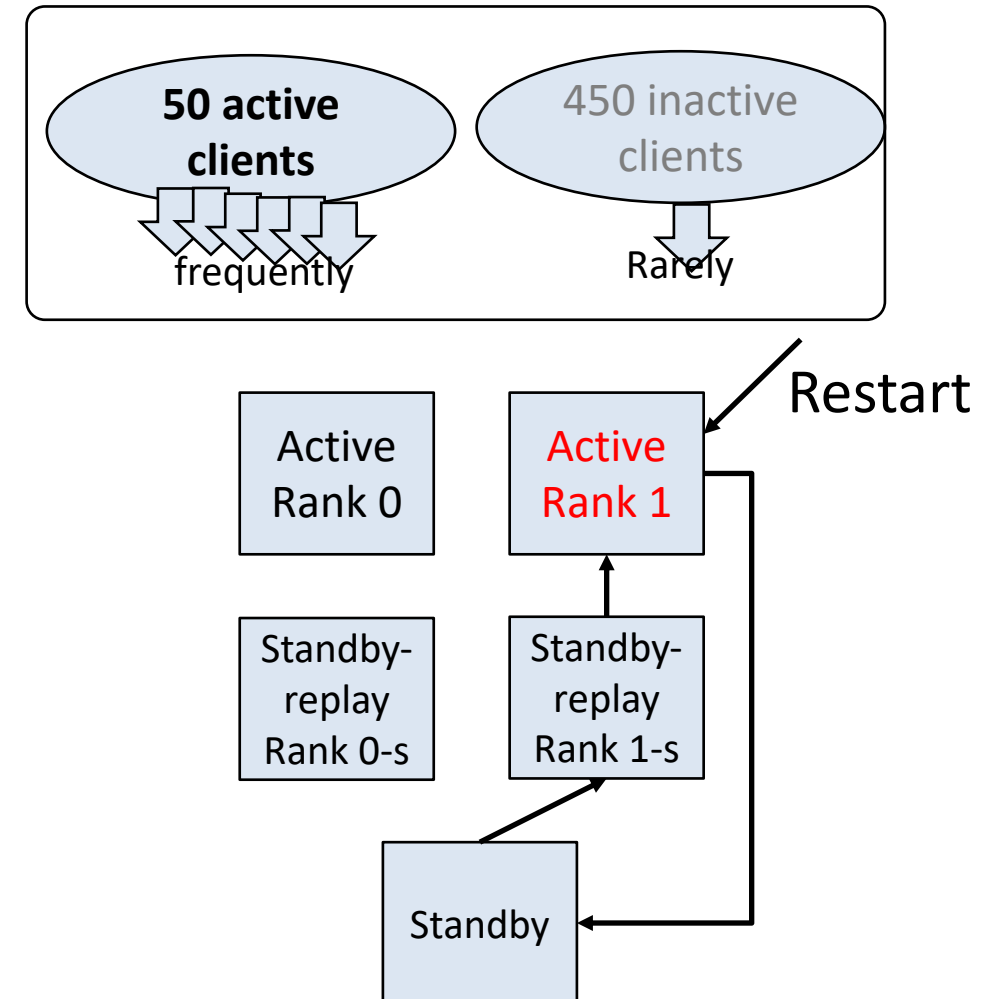


Subtree Pinning Pros/Cons

- Pros
 - Subtrees can clearly be isolated into specific MDS ranks
 - Improved cache locality results in the better performance
 - Exporting/importing subtrees among MDS ranks are prevented (e.g., minimizing balancing calculation and lock contention between MDS and clients)
- Cons
 - Load imbalance sometimes
 - Administrative burden → manually pinning (how?)
 - How do we know client workload dynamics?

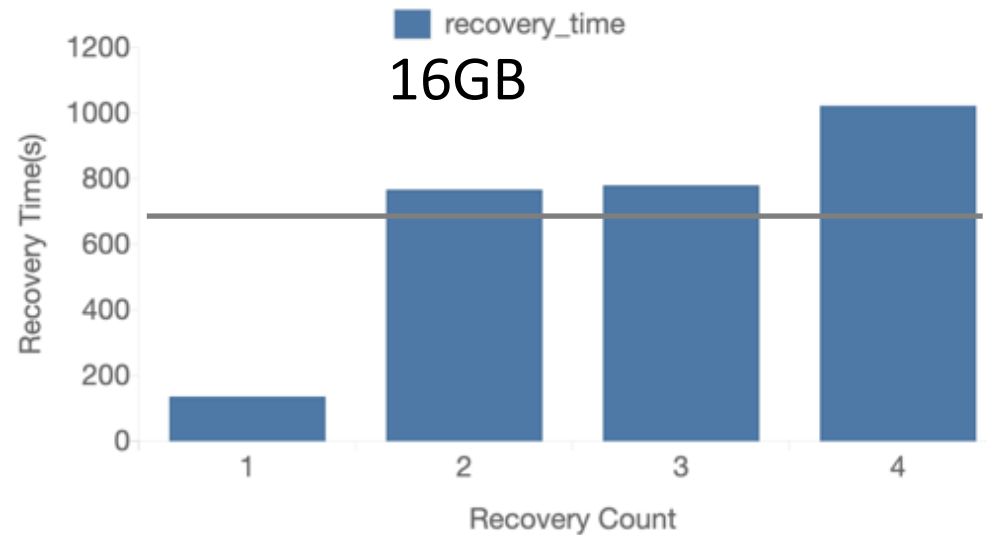
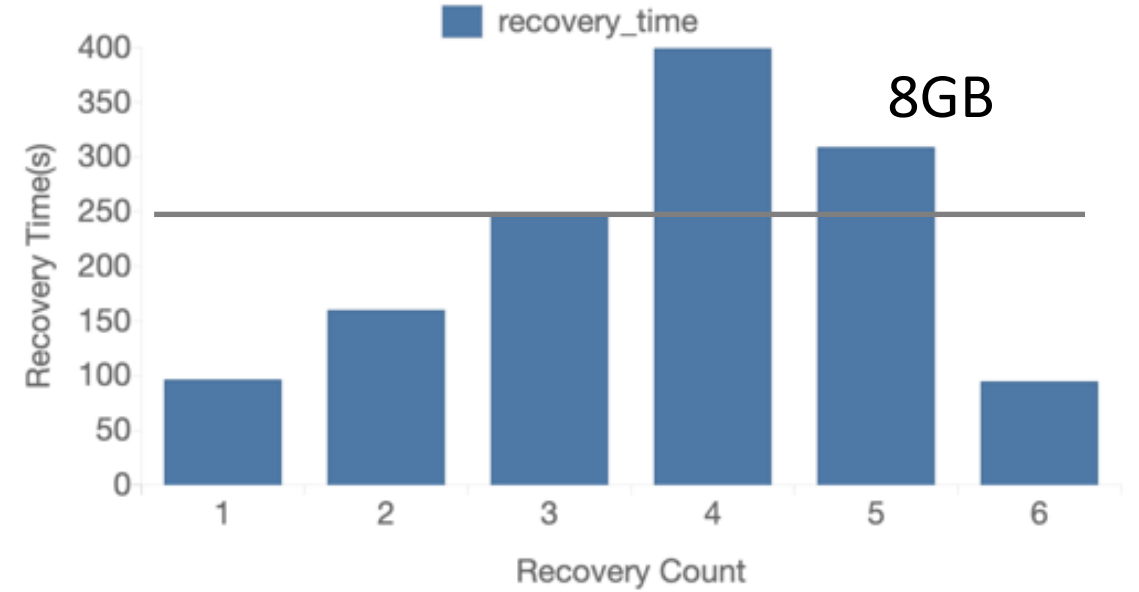
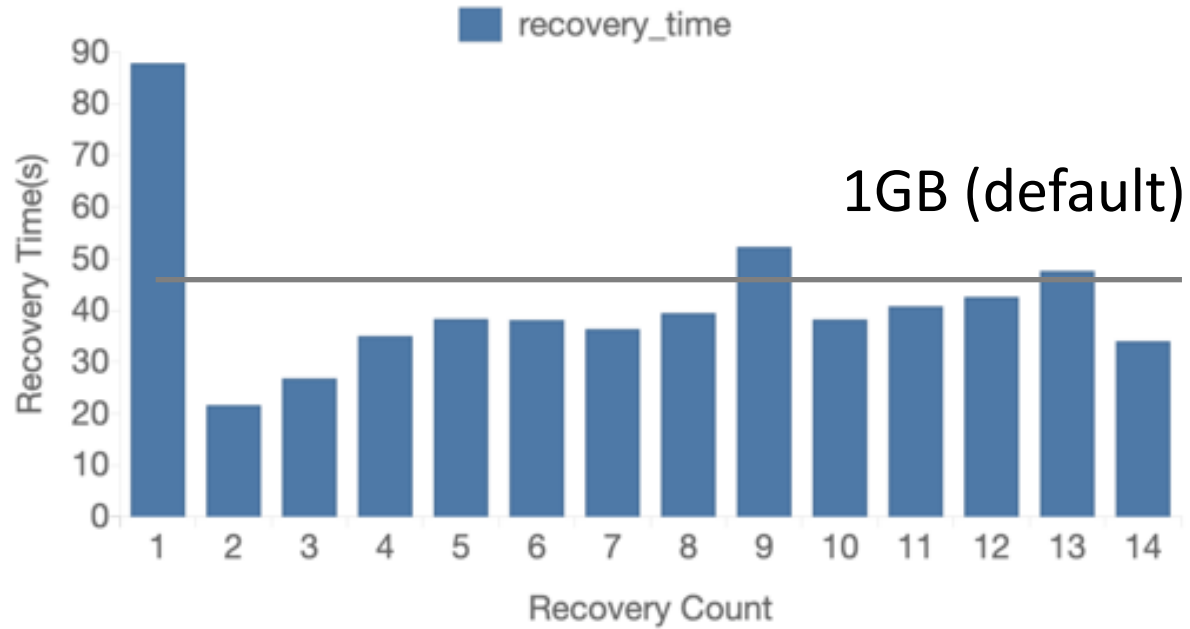
Test Configuration for MDS Recovery

Type	Spec
# of Active MDSs (max_mds)	2
Standby-replay MDSs	2
Standby MDS	1
mds_cache_memory_limit	1GB, 8GB, 16GB
# of Clients	50 active + (450 inactive)
CephFS Client	Kernel cephfs driver
Tool	vdbench
width	10
files	Relies on mds_cache_memory_limit (mds_cache_memory_limit * 350K * max_mds / width / num_clients)
Elapsed	10800
Operation Type	create, getattr, unlink
Thread	1



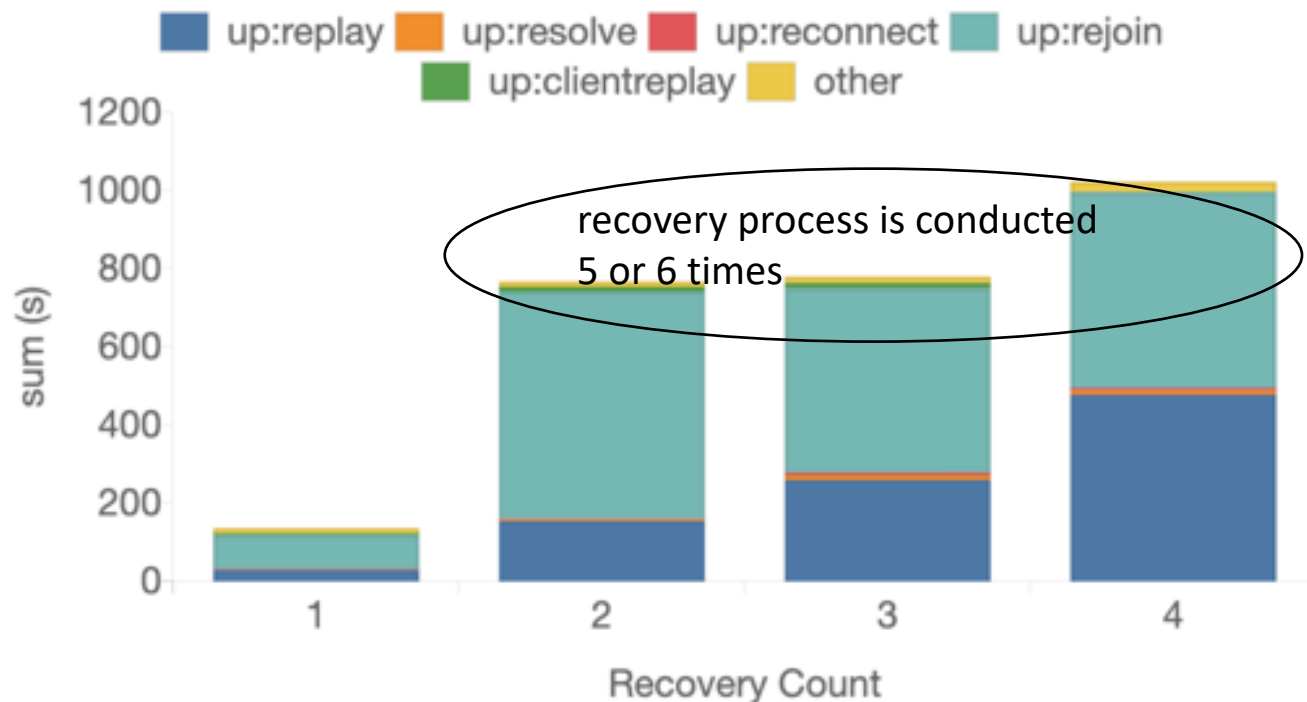
We measure the time when # of Active MDSs is changed from 1 to 2

Recovery Test Results



Recovery Time Breakdown for 16GB Cache

- “up:rejoin” takes up most of the recovery time
- In up:rejoin, the MDS is rejoining the MDS cluster cache



Skipping beacon heartbeat to monitors (last acked 50.0168s ago); MDS internal heartbeat is not healthy!

skipping upkeep work because connection to Monitors appears laggy

mds.mds003 respawn!

[Workaround]

mds_heartbeat_grace = 15s → larger value

ref.: <https://docs.ceph.com/en/latest/cephfs/mds-states/>

<https://lists.ceph.io/hyperkitty/list/dev@ceph.io/thread/VFY3A6CLBYLJ3MZSVCQA2Q5BTMFSHHZD/>

Outline of Contents

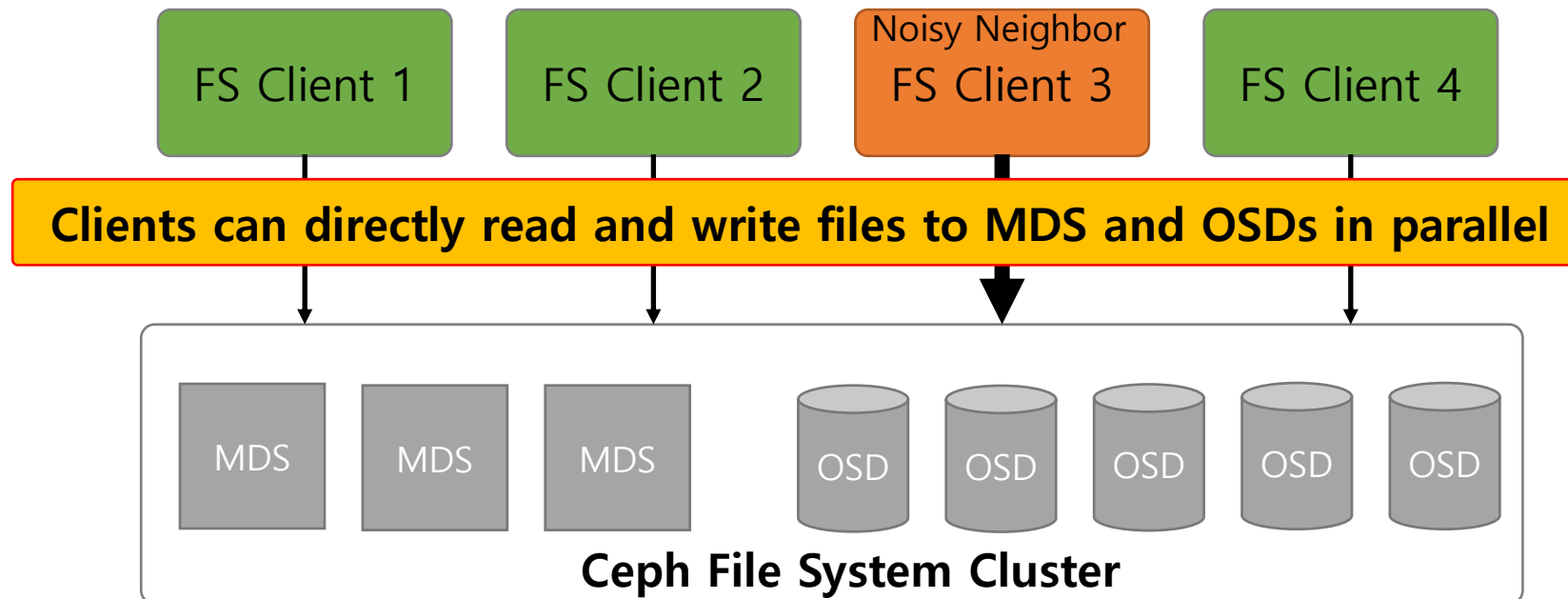
- CephFS Overview
- MDS Evaluation
 - MDS Scalability
 - CephFS Kernel vs FUSE clients
 - Impact of MDS Cache Size
 - Static Subtree Pinning
 - MDS Recovery Time
- **mClock QoS Scheduler for CephFS**
- Summary & Wrap Up

Storage Comparisons

	Amazon EFS	NetApp NAS	CephFS
Storage Backend	NFS	NFS	Ceph
Scalability	High	High	Very High
Reliability	✓	✓	✓
Capacity Quota	✓	✓	✓
Performance QoS	✓	✓	Planned

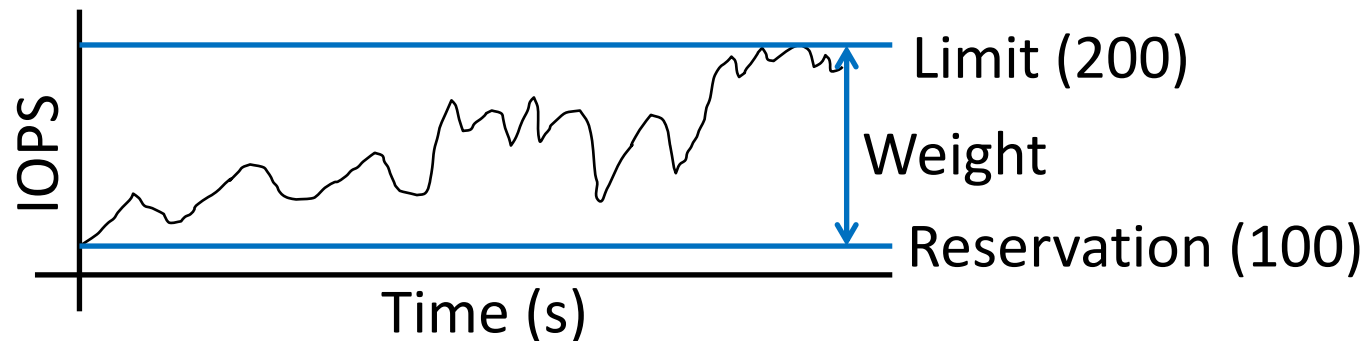
Technical Challenges

- Storage front-end doesn't exist
 - No gateways between clients and servers for scalability
 - Noisy neighbor problem happens inevitably



dmClock Scheduler Algorithm

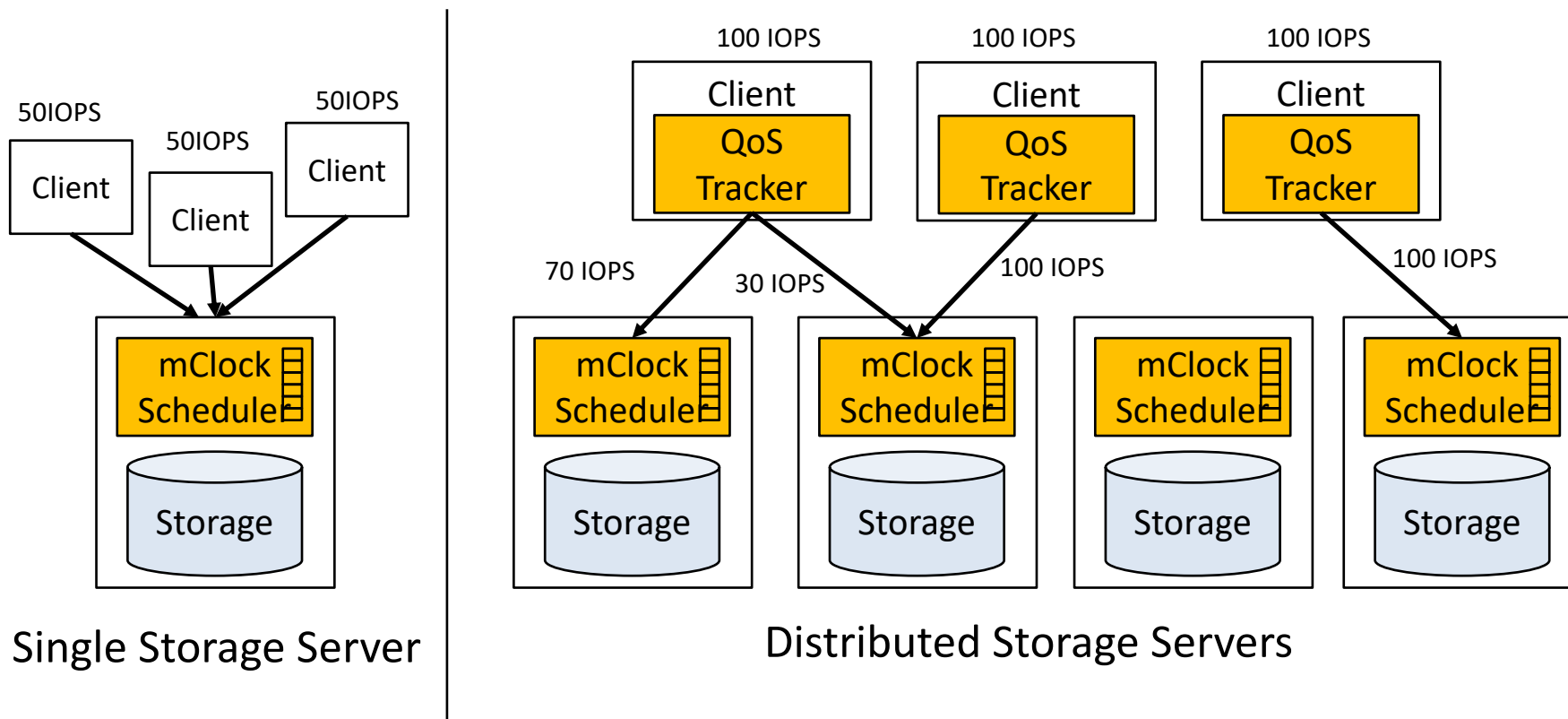
- The dmClock paper[1] has been published at USENIX OSDI'10 (developed by VMware)
- Three control knobs
 - **Reservation**: minimum guarantee
 - **Limit**: maximum allowed
 - **Weight**: proportional allocation
- Features of dmClock
 - Supports all controls in a single algorithm
 - Easy to implement
 - Library in Ceph (src/dmclock)



[1] "mClock: Handling Throughput Variability for Hypervisor I/O Scheduling", Ajay Gulati et al., USENIX OSDI, 2010

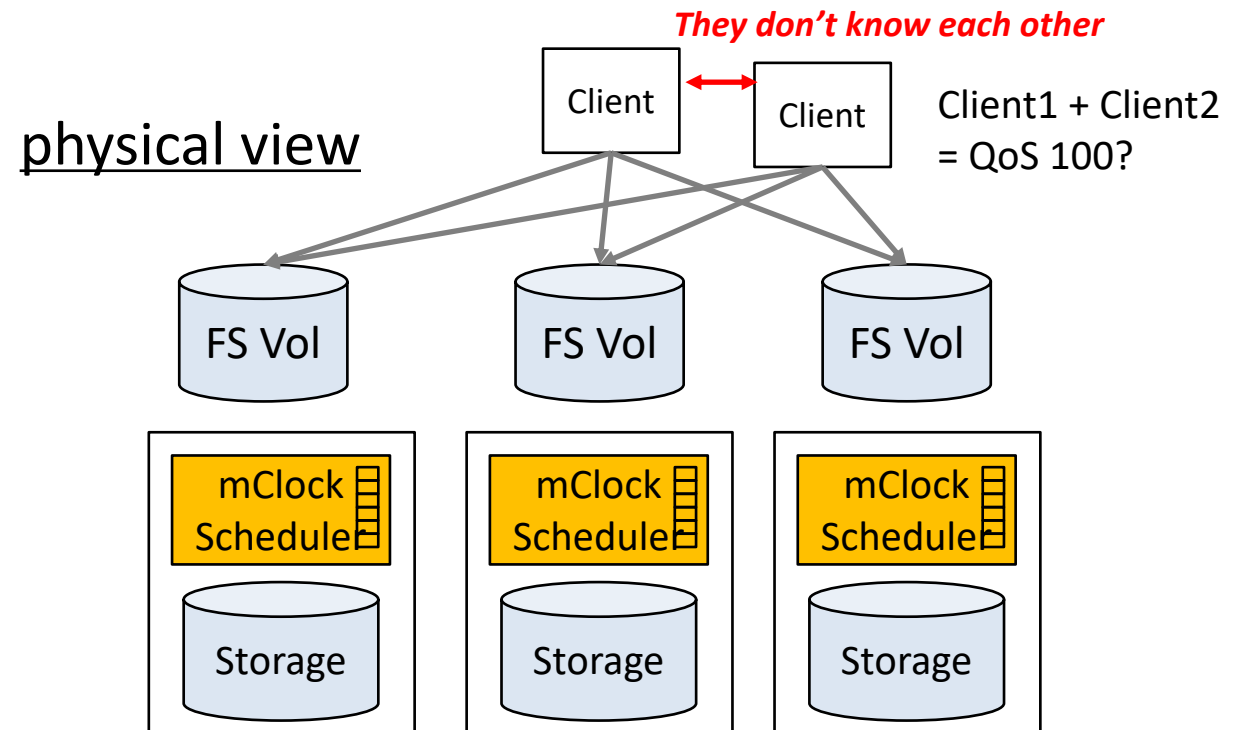
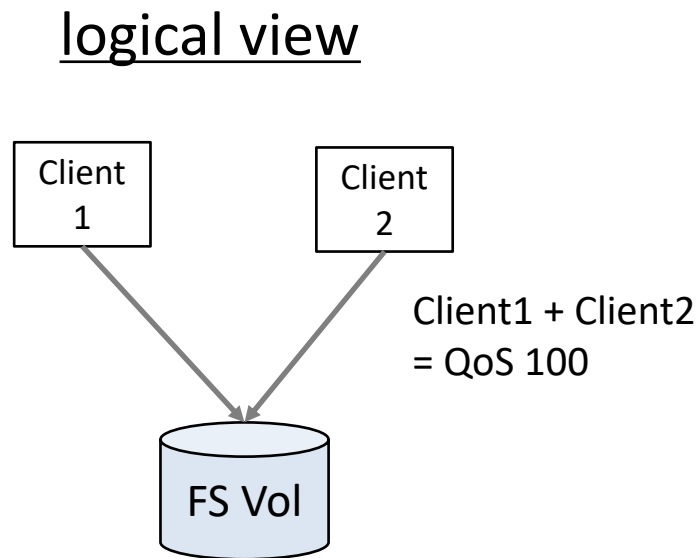
dmClock Scheduler Architecture

- Distributed mClock model
 - Each server maintains mClock scheduler
 - Each QoS tracker in client manages how many requests have been done previously



Limitations of dmClock for CephFS QoS

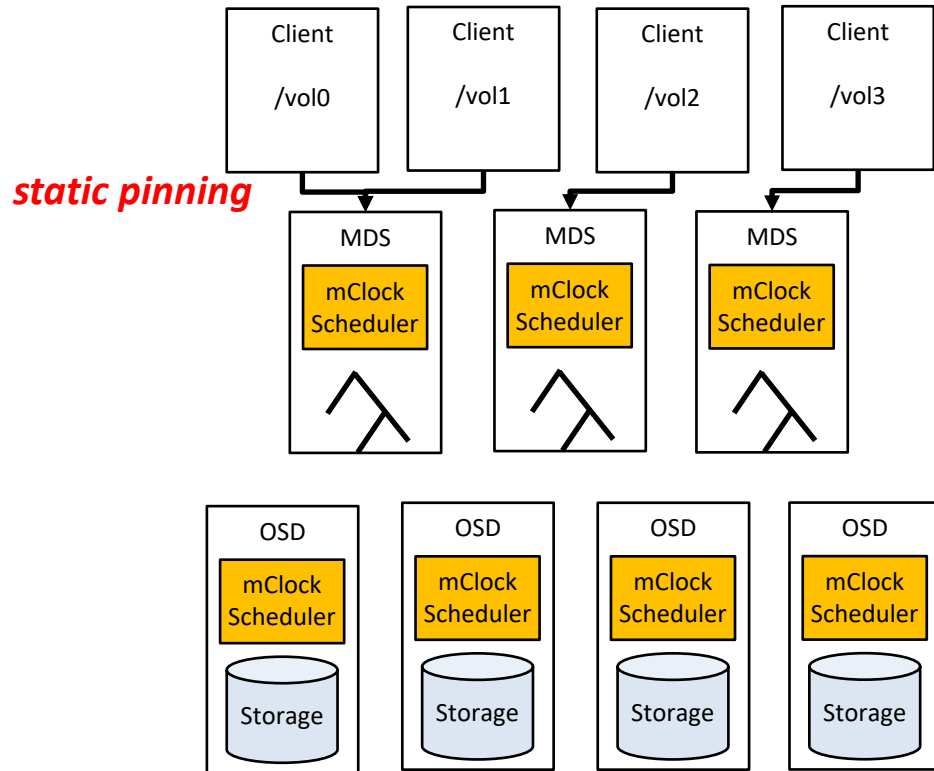
- **Modifications to the client SW stack**
 - Backward compatibility issue → what if the client disables the QoS feature
- **Multiple different clients on the same resource (e.g., volume)**
 - dmClock doesn't consider the volume share case



CephFS QoS Solutions

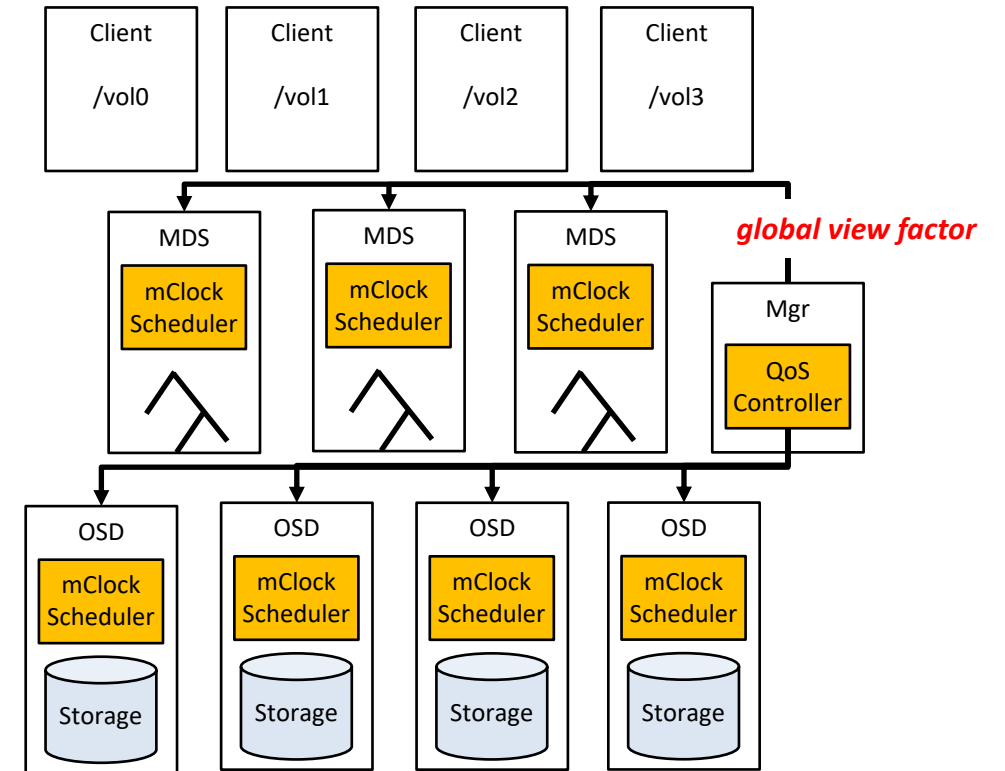
mClock + Static Pinning

Our Approach



Global QoS Controller Solution

KAIST Research Project



Ceph Community Proposal


- <https://lists.ceph.io/hyperkitty/list/dev@ceph.io/thread/XO33ZPJ3BONNIKWMGN6A7K62F74C5AJO/>
 - Feedbacks from maintainers, Gregory and Josh

← 이전

[RFC] CephFS dmClock QoS Scheduler

Seeking in

Next nautilus point release...


 Yongseok Oh

Hi Ceph maintainers and developers,

The objective of this is to discuss our work on a dmClock based client QoS management for CephFS.

Our group at LINE maintains Ceph storage clusters such as RGW, RBD, and CephFS to internally support OpenStack and K8S based private cloud environment for various applications and platforms including LINE messenger. We have seen that the RGW and RBD services can provide consistent performance to multiple active users since RGW employs the dmClock QoS scheduler for S3 clients and hypervisors internally utilize I/O throttler for VM block storage clients. Unfortunately, unlike RGW and RBD, CephFS clients can directly issue metadata requests to MDSs and filedata requests OSDs as they want. This situation occasionally (or frequently) happens and the other client performance may be degraded by the noisy neighbor. In the end, consistent performance cannot be guaranteed in our environment. From this observation and motivation, we are now considering the client QoS scheduler using the dmClock library for CephFS.

A few things about how to realize the QoS scheduler.


 Josh Durgin

Hello Yongseok,

That is a very exciting project! We are also interested in improving QoS. For Pacific, we've been focusing on OSD-side mclock and balancing background work vs clients.

Sridhar is doing extensive testing and fixed a few issues with the op scheduler integration and dmclock library, e.g.:

<https://github.com/ceph/ceph/pull/37031>
<https://github.com/ceph/ceph/pull/37431>

 Gregory Farnum

On Mon, Sep 28, 2020 at 9:26 PM Josh Durgin <jdurgin@redhat.com> wrote:
| ...

That's basically what I've got. CephFS is tricky because we naively expect the clients to be scaling out as well; most single-mounter scenarios just end up on rbd because it's simpler.

MDS mClock QoS Scheduler PR

- <https://github.com/ceph/ceph/pull/38506> (under review)
 - Coauthored by Jinmyeong Lee
- Key features
 - Per subvolume MDS QoS
 - QoS configuration through asok command
 - Virtual extended attributes for each subvolume
- Testing
 - Teuthology test (w qa/tasks/cephfs/test_* + test_mds_dmclock_qos)
 - Unittest (w bin/unittest_mds_dmclock_sched)
 - Performance (w smallfile)

Example of MDS QoS Configuration

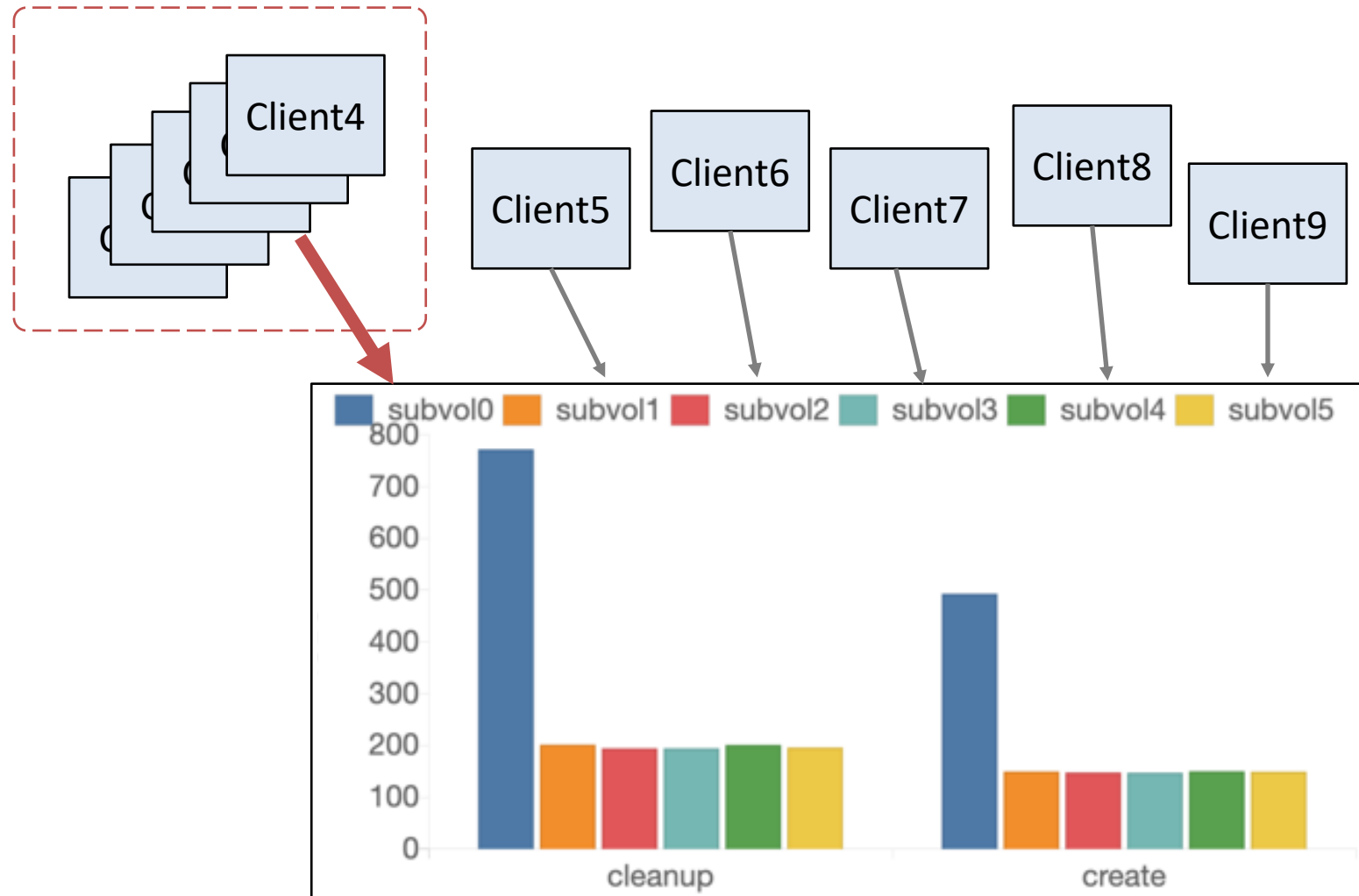
- Default configuration through asok

```
ceph --admin-daemon /var/run/ceph/mds.a.asok config set mds_dmclock_enable true
ceph --admin-daemon /var/run/ceph/mds.a.asok config set mds_dmclock_reservation 1000
ceph --admin-daemon /var/run/ceph/mds.a.asok config set mds_dmclock_weight 1500
ceph --admin-daemon /var/run/ceph/mds.a.asok config set mds_dmclock_limit 2000
```

- Per subvolume configuration with vxattr

```
ceph-fuse -n client.admin -c ceph.conf -k keyring --client-mountpoint=/mnt
setfattr -n ceph.dmclock.mds_reservation -v 1000 /mnt/volumes/_nogroup/subvolume/
setfattr -n ceph.dmclock.mds_weight -v 1000 /mnt/volumes/_nogroup/subvolume/
setfattr -n ceph.dmclock.mds_limit -v 1000 /mnt/volumes/_nogroup/subvolume/
```

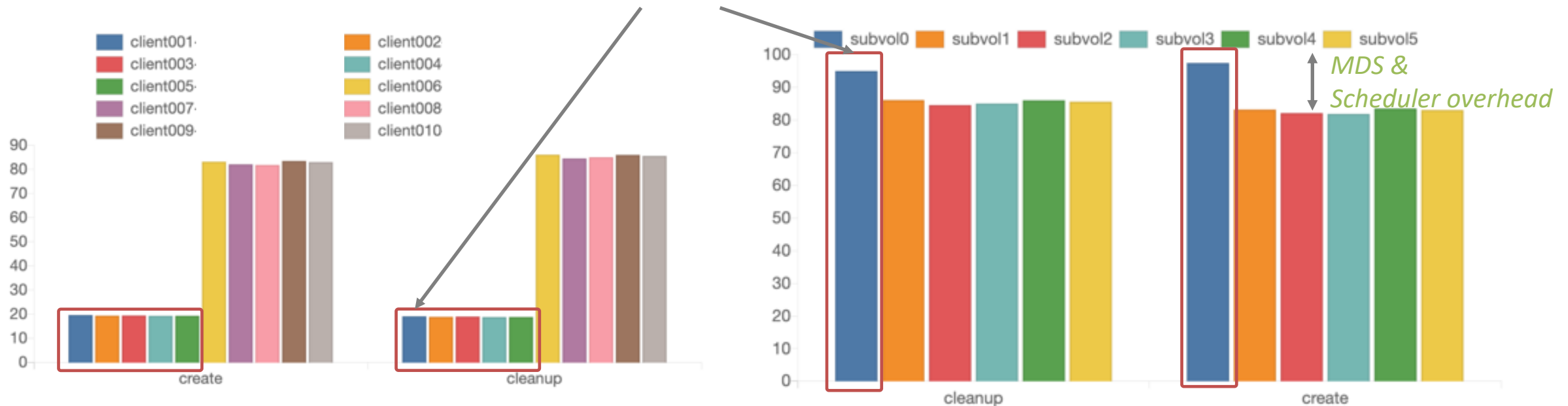
Noisy Neighbor Scenario



MDS mClock Preliminary Results

- mClock limit 100 with subtree pinning is applied
 - 12,000 files are created and deleted on each subvolume

Volume QoS resource is shared to 5 clients



Summary

- **CephFS Benefits**

- CephFS can be used as a backend fs for Manila and K8S PV
- Its stability and reliability has been enhanced recently
- Administrators can easily adopt CephFS along with RGW, and RBD in Ceph

- **Technical Challenges**

- MDS multi-core scalability
- Kernel driver provides superior performance, but OS rebooting is necessary if kernel issues happen
- MDS Performance is very sensitive to *mds_cache_memory_limit*
- Increased MDS cache size affects high availability due to recovery
- Subtree pinning can isolate some workloads to a specific rank, however, administrators should identify their characteristics before applying
- QoS feature is not supported

Revisiting CephFS MDS and mClock QoS Scheduler



Ceph Korea
Community Seminar
2021 04 14



LINE Cloud Storage
Yongseok Oh

- Thank you
- Q & A

References

- Ceph: A Scalable, High-Performance Distributed File System, Sage Weil, OSDI'06
- Overview and Status of the Ceph File System, Patrick Donnelly, 2018, <https://indico.cern.ch/event/644915/>
- CephFS with OpenStack Manila based on BlueStore and Erasure Code, 유장선, 2018, <https://cutt.ly/dc7Qnn7>
- ceph-linode for CephFS testing, Patrick Donnelly, <https://github.com/batrack/ceph-linode>