Revisiting CephFS MDS and mClock QoS Scheduler

Ceph Korea
Community Seminar
2021 04 14

LINE Cloud Storage
Yongseok Oh

Outline of Contents

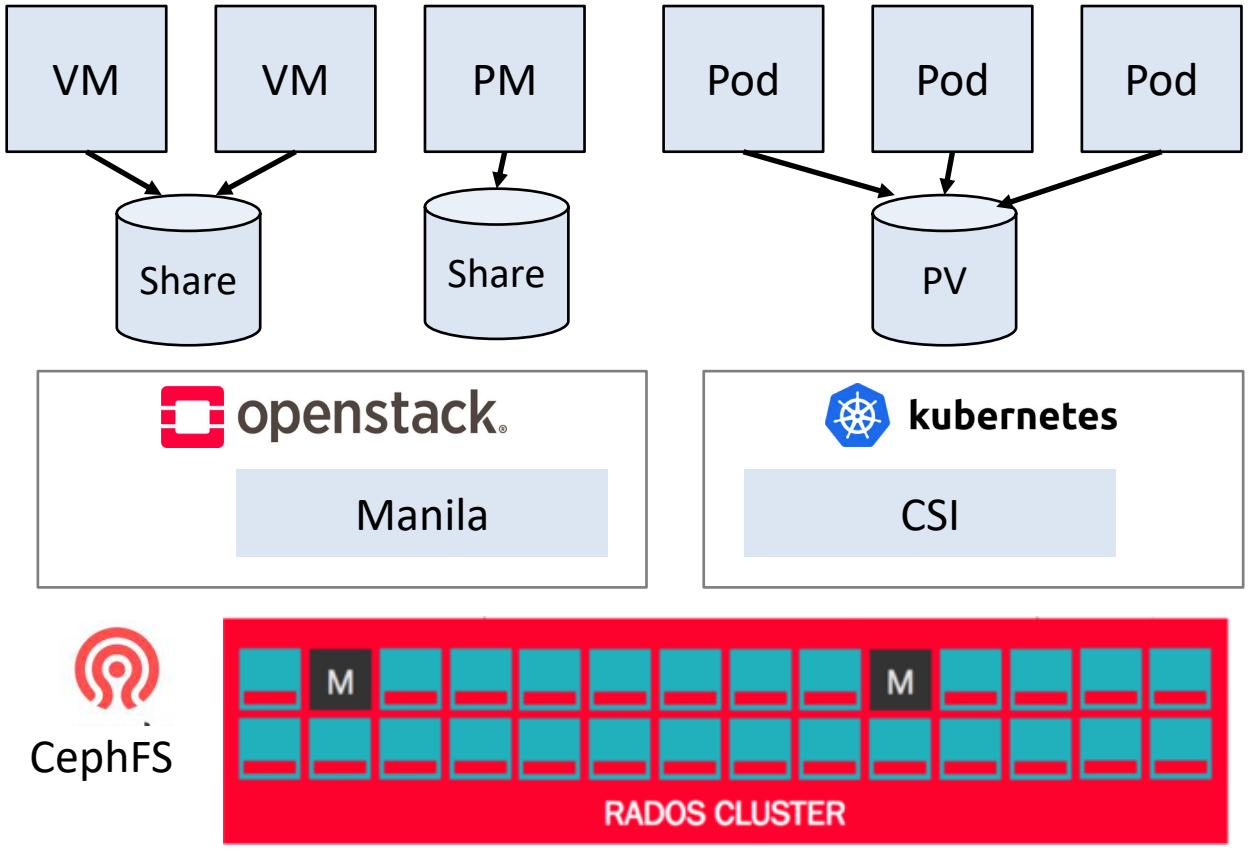
- CephFS MDS Overview
- MDS Evaluation
  - MDS Scalability
  - CephFS Kernel vs FUSE clients
  - Impact of MDS Cache Size
  - Static Subtree Pinning
  - MDS Recovery Time
- mClock QoS Scheduler for CephFS
- Summary

Key Features of CephFS

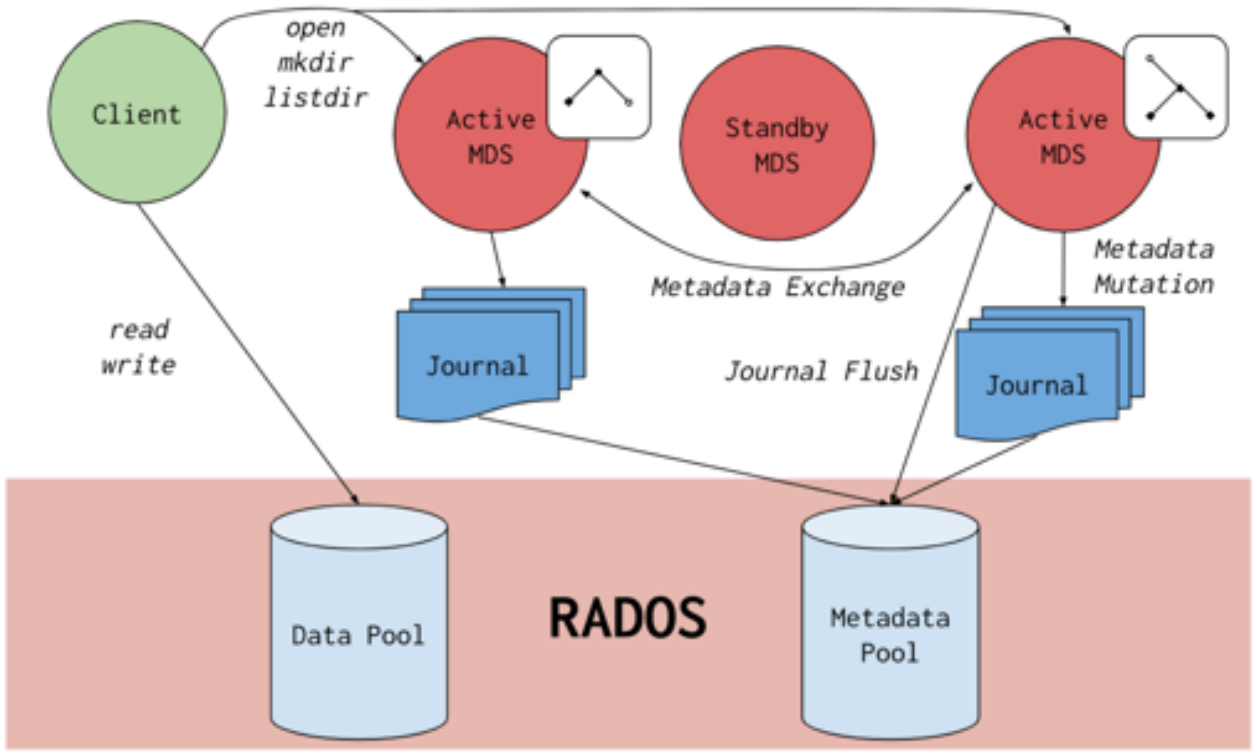
- POSIX compliant distributed file system
- Multi active MDSs (e.g., scalability)
- Standby/standby-replay MDSs (e.g., rapid failover)
- Journaling (e.g., guaranteeing metadata consistency)
- Dynamic/static sub directory partitioning
- Kernel/FUSE/libcephfs client support
- Subvolume/snapshot/quota management
- 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



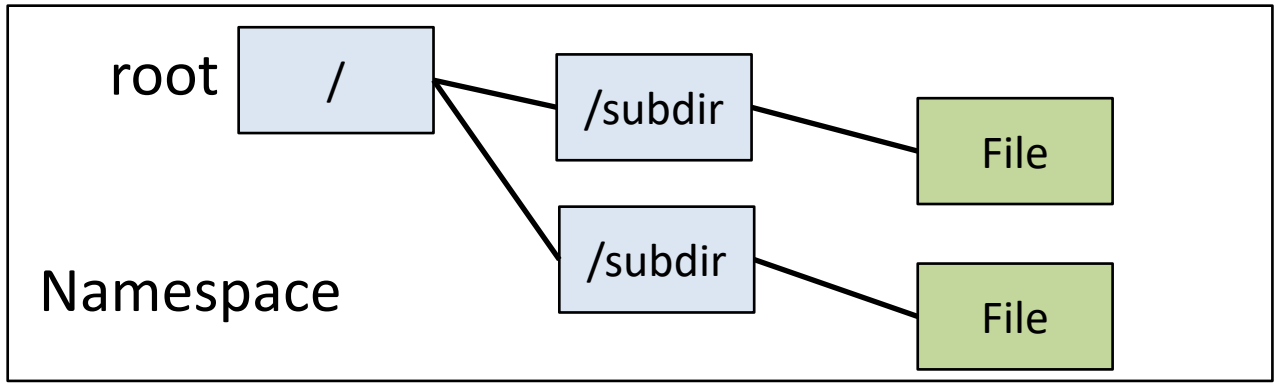
MDS Main Role

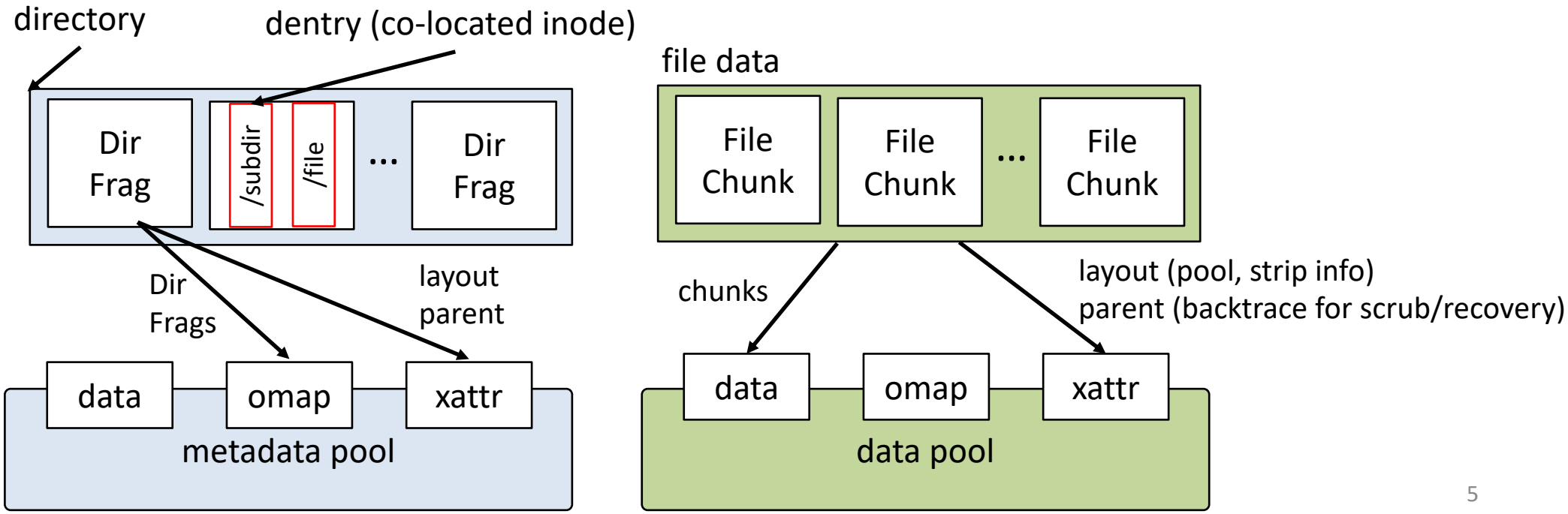
/Dir/Filename

MDS Path Translation

Object ID

MDS Data Structures





MDS Object Types

Type	Obj Name	MDS Ino	Description	Omap	Data
root	1.00000000	0x1	Root directory	V	
MDSDIR	100.00000000	0x100 ~	Stray directory ~mds0	V	
Log	200.00000000	0x200 ~	Log event		V
Log Backup	300.00000000	0x300 ~	Log event backup		V
Log Pointer	400.00000000	0x400 ~	Log pointer		V
Purge Queue	500.00000000	0x500 ~	Purge queue journal		V
Stray	600.00000000	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	10000000001.00000000	0x10000000001~0x1FFFFFFFFF	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 00  |........I.......|
00000010  00 00 00 00 01 00 00 00  00 00 00 30 39 75 60 fb  |...........09u`.|
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 81  |..........09u`..|
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

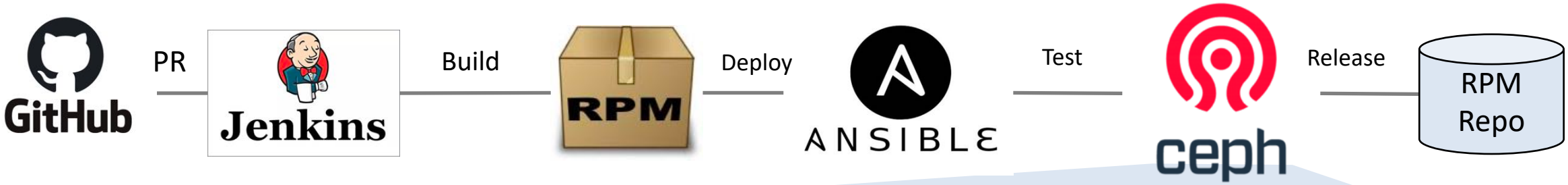
- CephFS MDS Overview
- **MDS Evaluation**
  - **MDS Scalability**
  - **CephFS Kernel vs FUSE clients**
  - **Impact of MDS Cache Size**
  - **Static Subtree Pinning**
  - **MDS Recovery Time**
- mClock QoS Scheduler for CephFS
- Summary & Wrap Up

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.

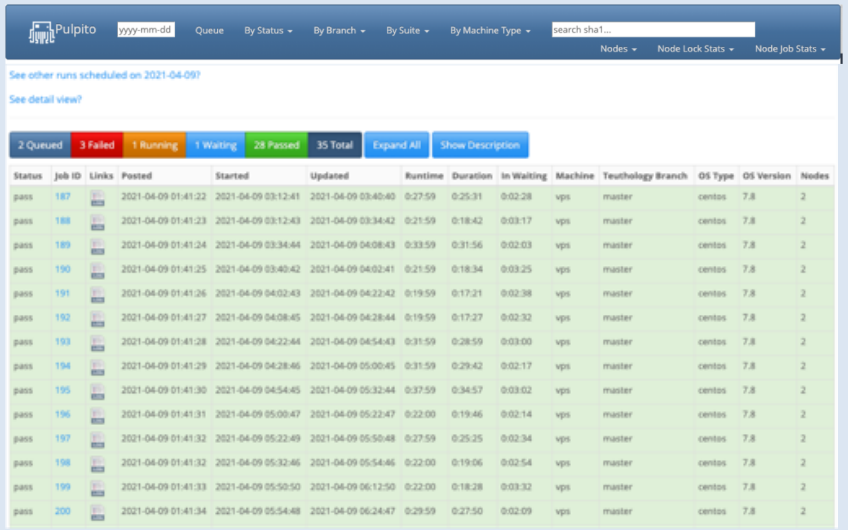
Type	Spec	Count
MDS	4 * vCPU 32GB RAM	16
OSD	40 * pCPU 128GB RAM	9 Servers * 6 SATA SSDs = 54
Mon	2 * vCPU 4GB RAM	3
Mgr	4 * vCPU 8GB RAM	3
Client	4 * vCPU 8GB RAM	20
Network	10Gbps	

Ceph CI/CD System in LINE

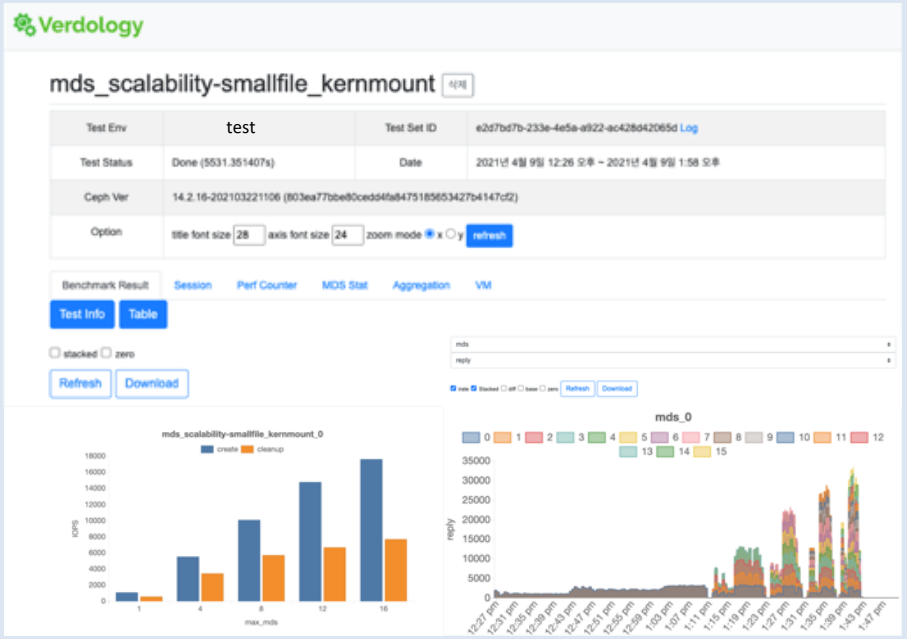


Ceph Test Frameworks

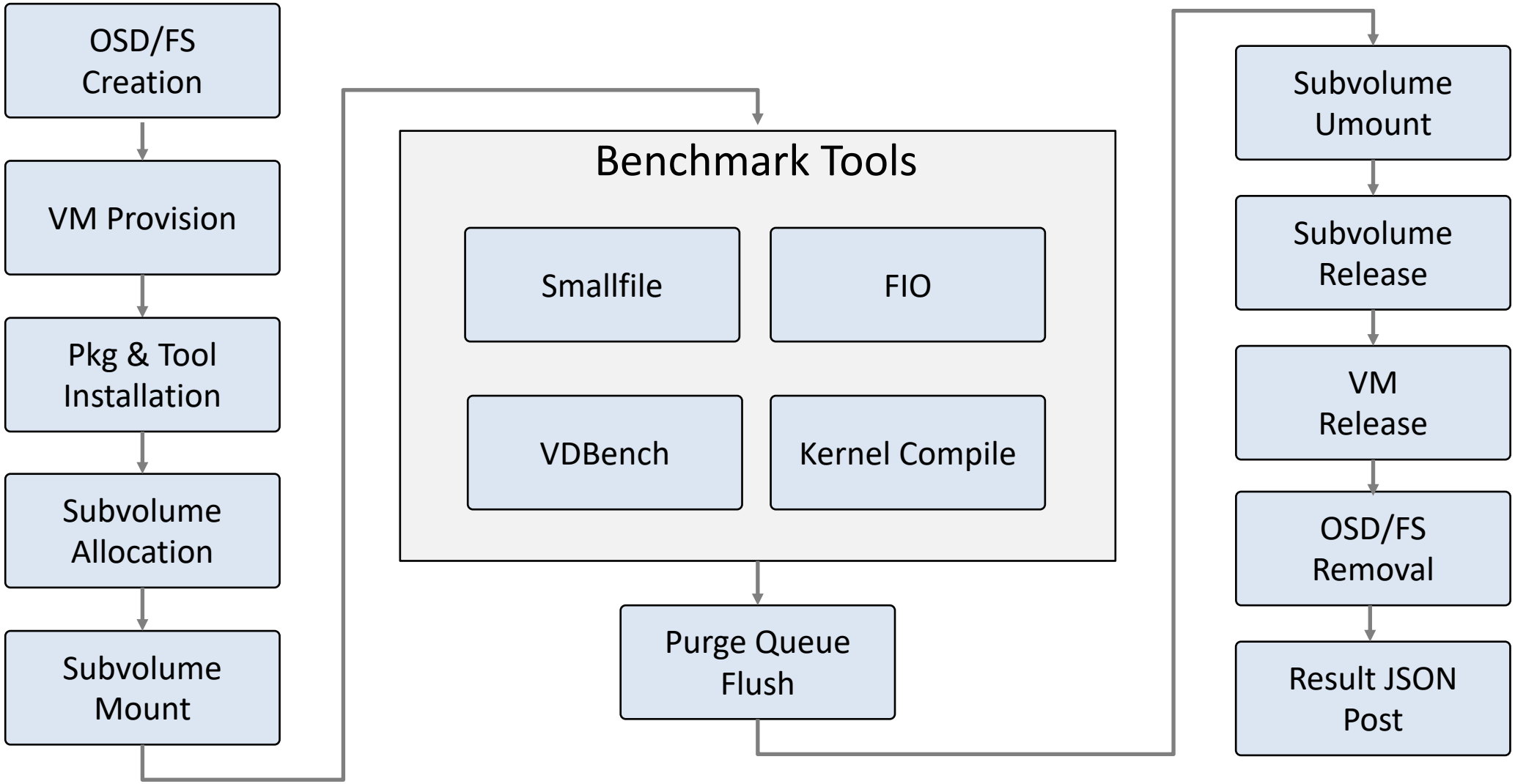
[teuthology]



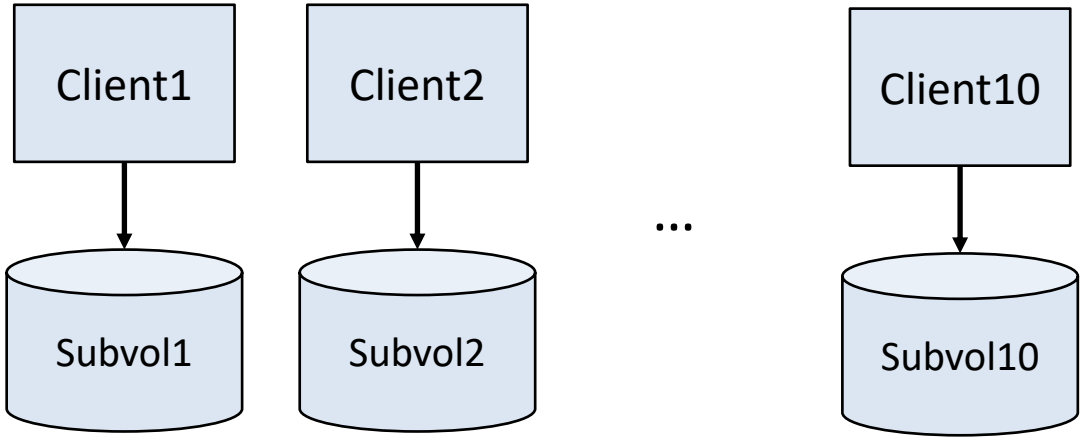
[verdology]



CephFS Test Flow of Verdology



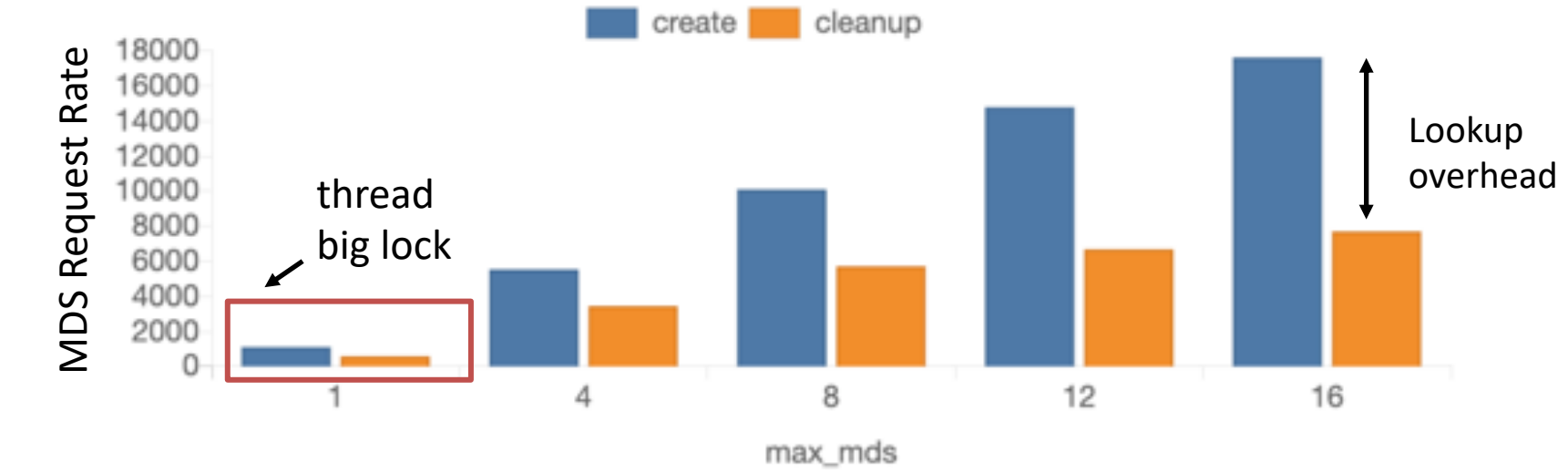
Benchmark Configuration for MDS Scalability

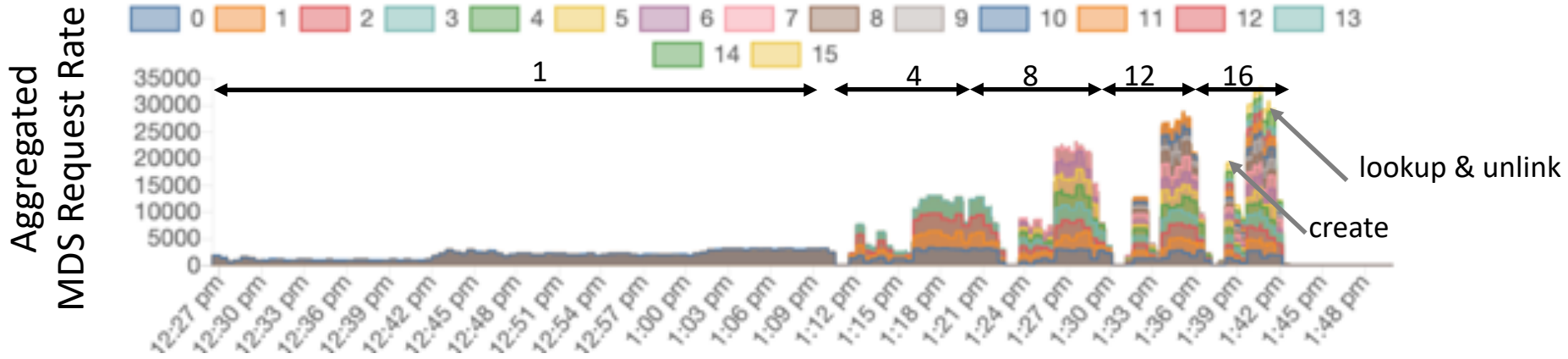


Each client is working on its own subvolume.

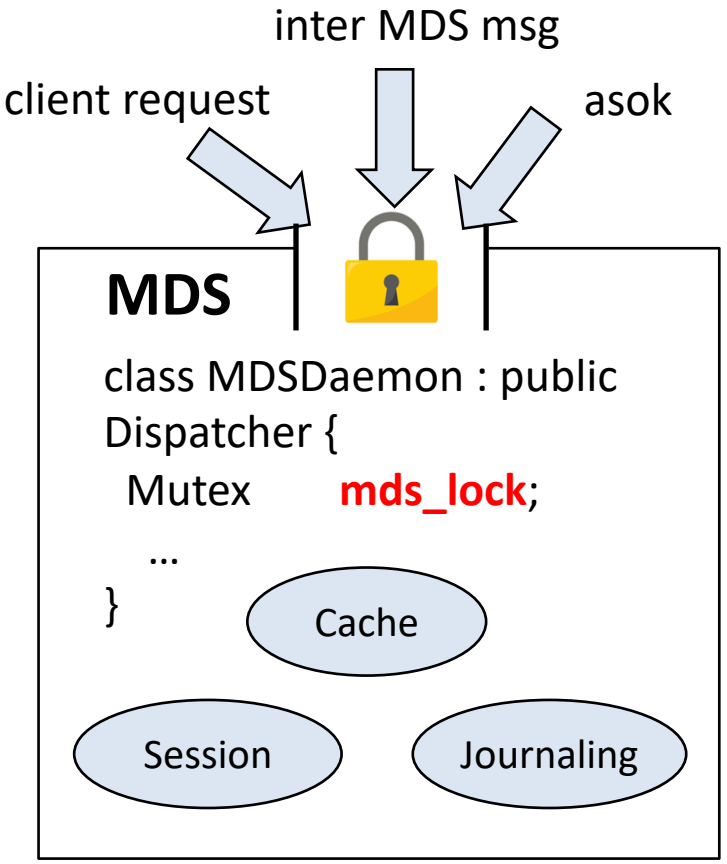
Type	Spec	Remark
Tool	Smallfile	
max_mds	1~16	
# of Clients	10	
Files per Client	50,000	
File Size	0	
Files per Dir	1000	
Operation Type	Create, Unlink	
Thread	1	
CephFS Client	Kernel cephfs driver	

MDS Scalability Results

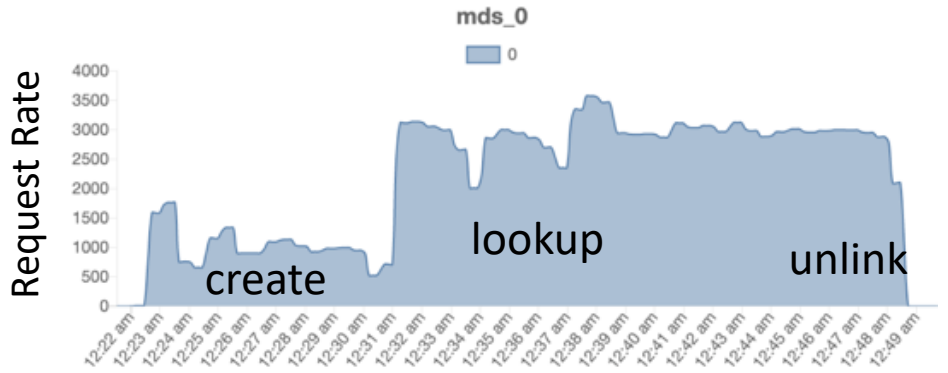




Less Scalable with Big Lock




```
  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
```



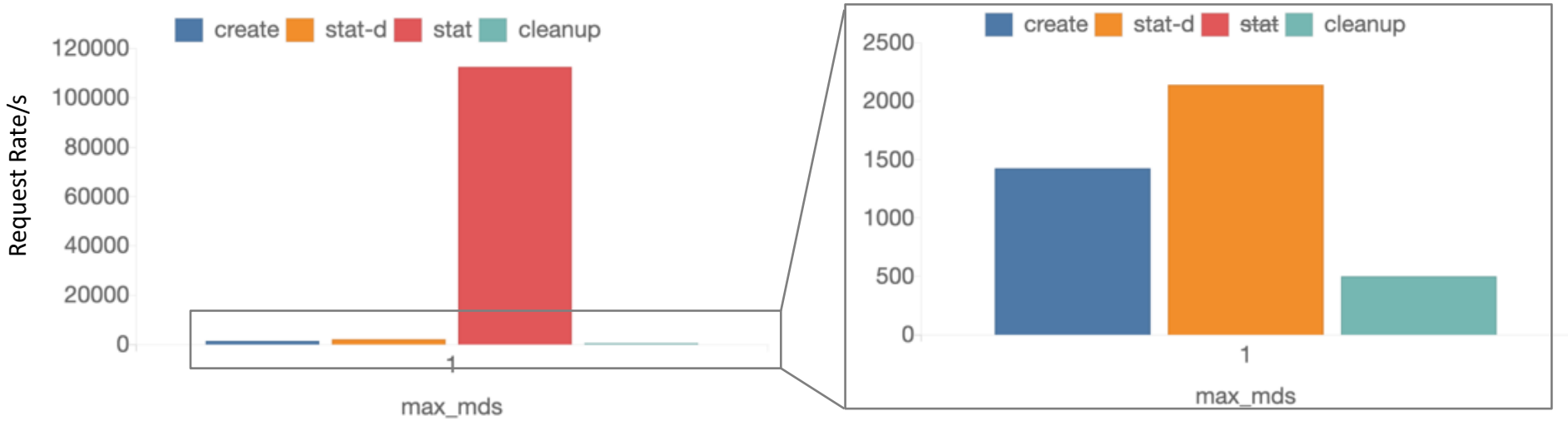
Backlog https://trello.com/c/U575CFEG/293-mds-multithreading

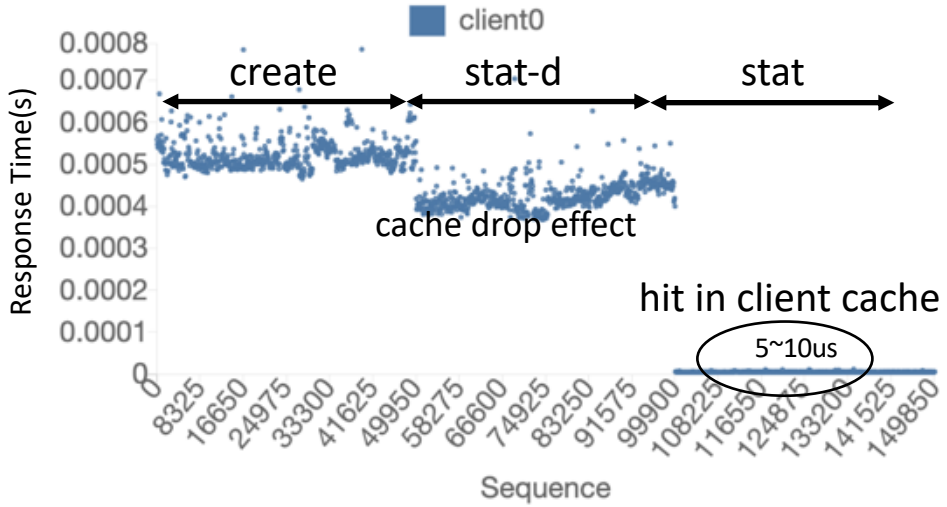
Test Parameters for Comp. of Kernel and FUSE

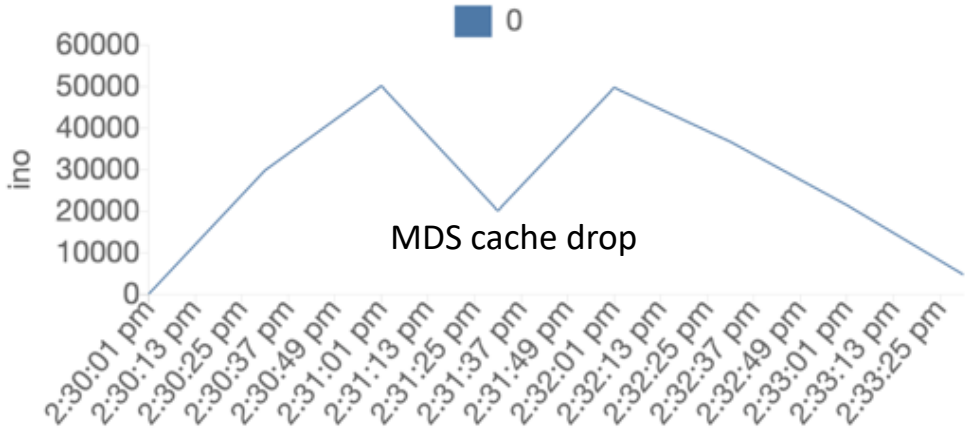
Type	Spec	Remark
# of Active MDSs (max_mds)	1	
# of Clients	**1**	
CephFS Client	Kernel cephfs driver, ceph-fuse	
Tool	Smallfile	
Files per Client	50,000	
File Size	0	
Files per Dir	1000	
Operation Type	Create, Stat-d, Stat, Unlink	
Thread	1	

- 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

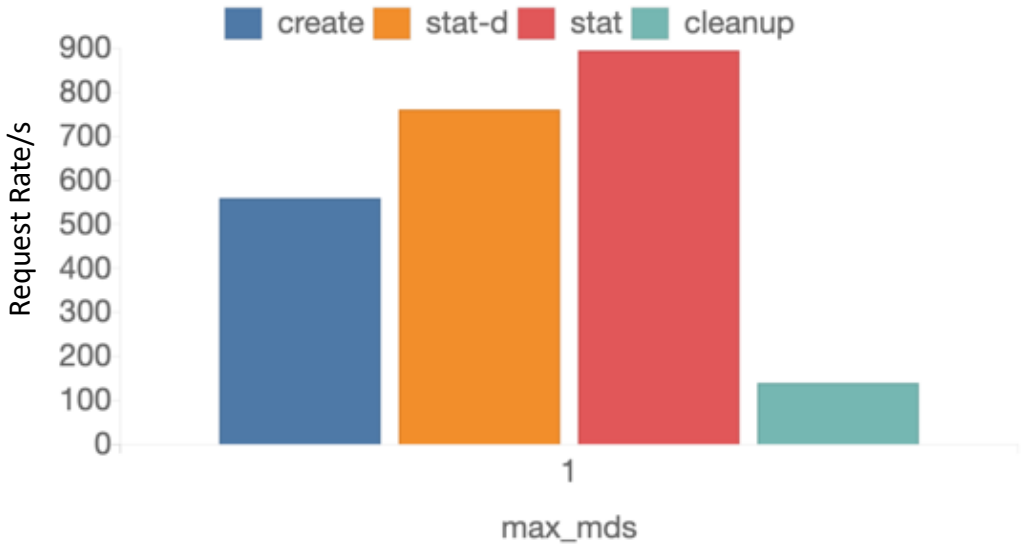
Kernel based Client



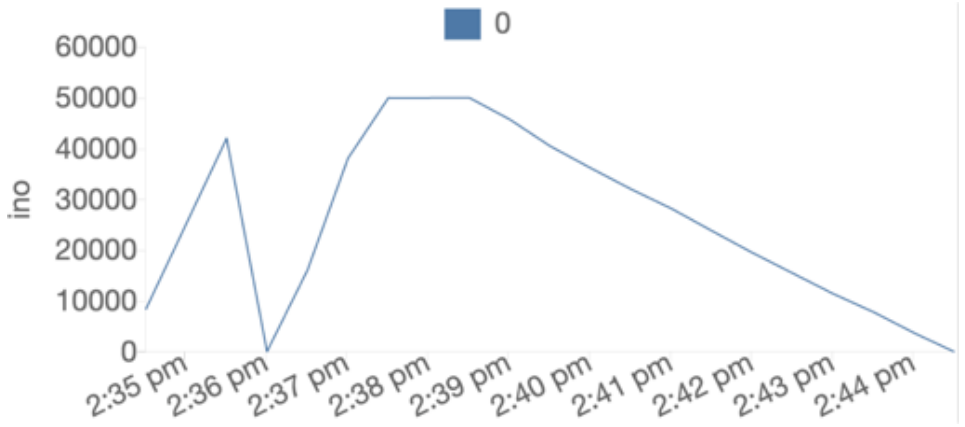




FUSE based Client

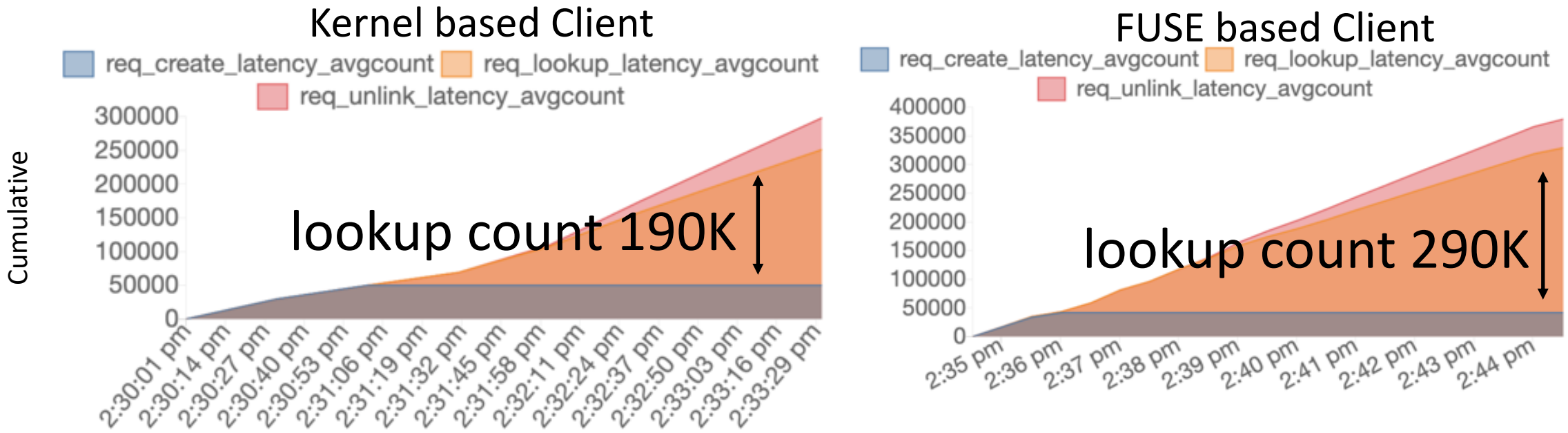


FUSE Client Configuration
client_cache_size = 16384 (by default)



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



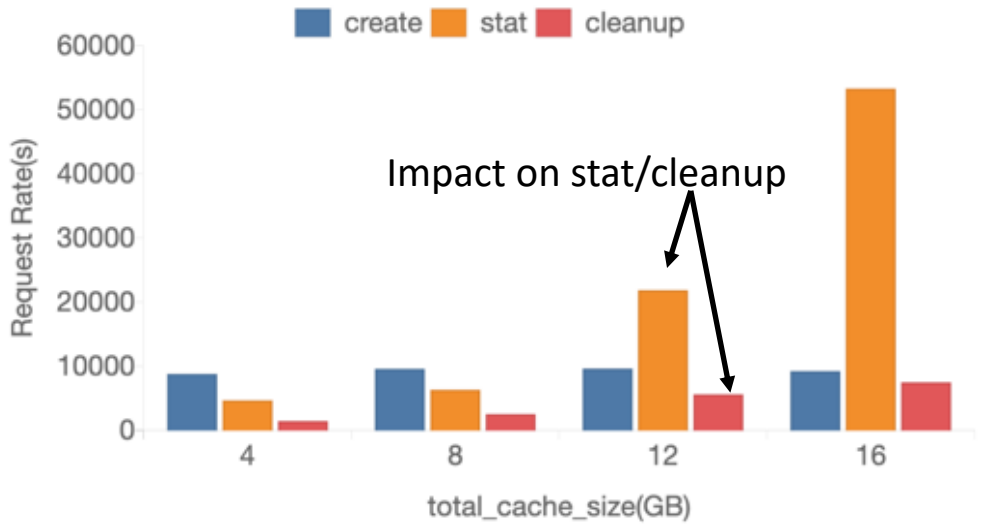
For simplicity and quick experiments, we have utilized kernel CephFS driver unless mentioned otherwise!

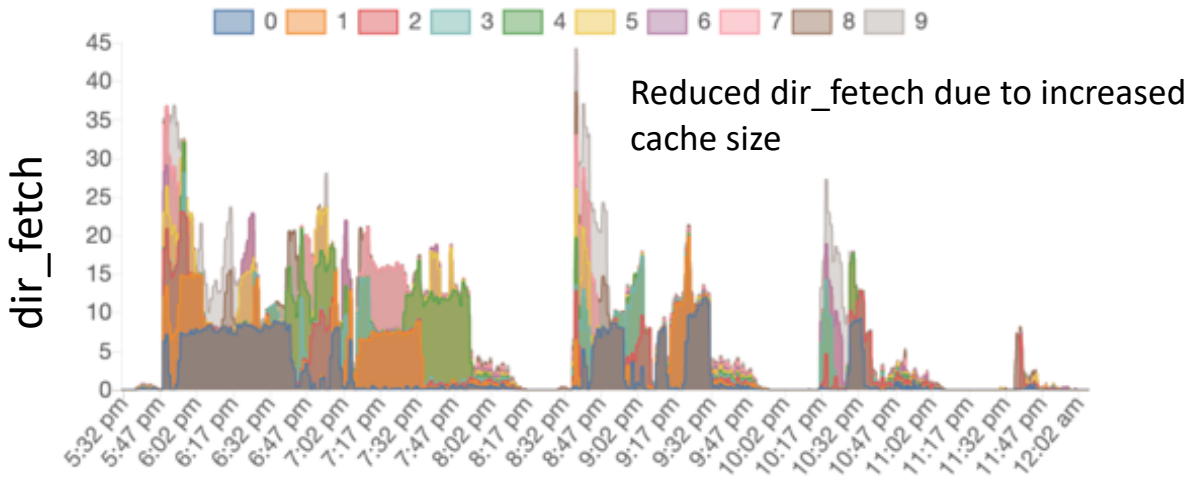
Test Parameters for MDS Cache Size

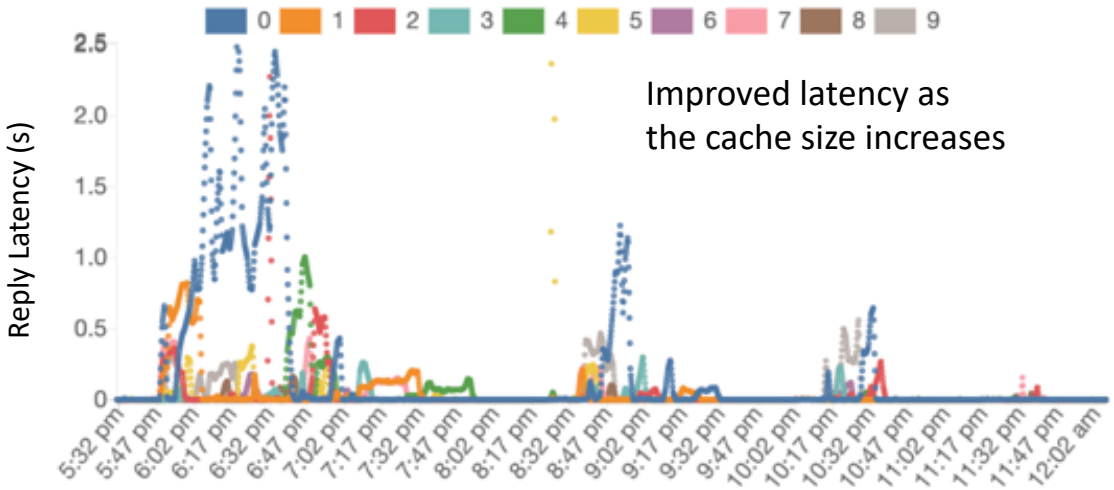
Type	Spec		Remark
# of Active MDSs (max_mds)	10		
Total_cache_size for all MDSs	4GB, 8GB, 12GB, 16GB (Example, if the total cache size is 4GB, each MDS has 4GB/10 MDSs)		
# of Clients	**20**		
CephFS Client	Kernel cephfs driver		
Tool	Smallfile	Kernel compile	
Files per Client	200K = 4Mill files / 20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

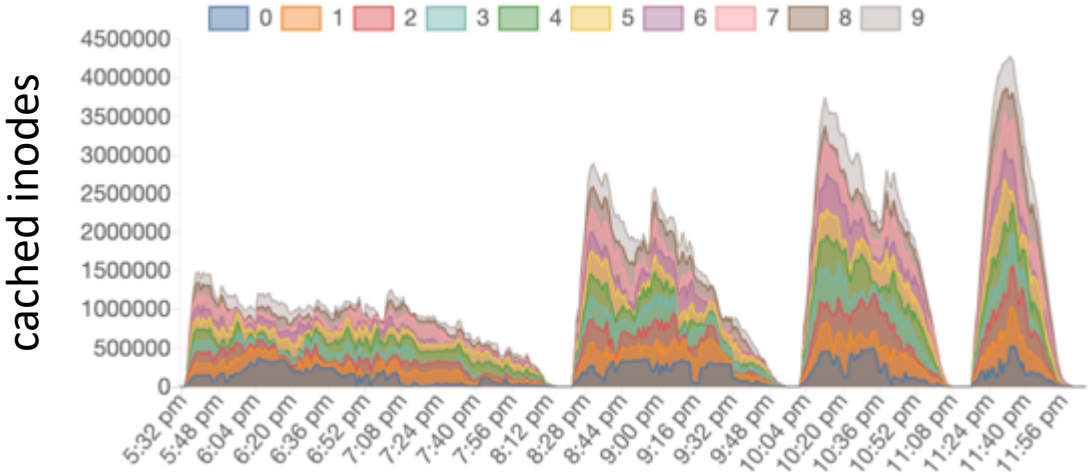
- Assumption
  - 350K inodes (including dentries) per GB (mds_cache_memory_limit)
  - 16GB MDS caches can retain more than 4Mill inodes approximately
  - To analyze impact of cache size, we vary the total MDS size from 4 to 16GB

Impact of MDS Caches (smallfile)



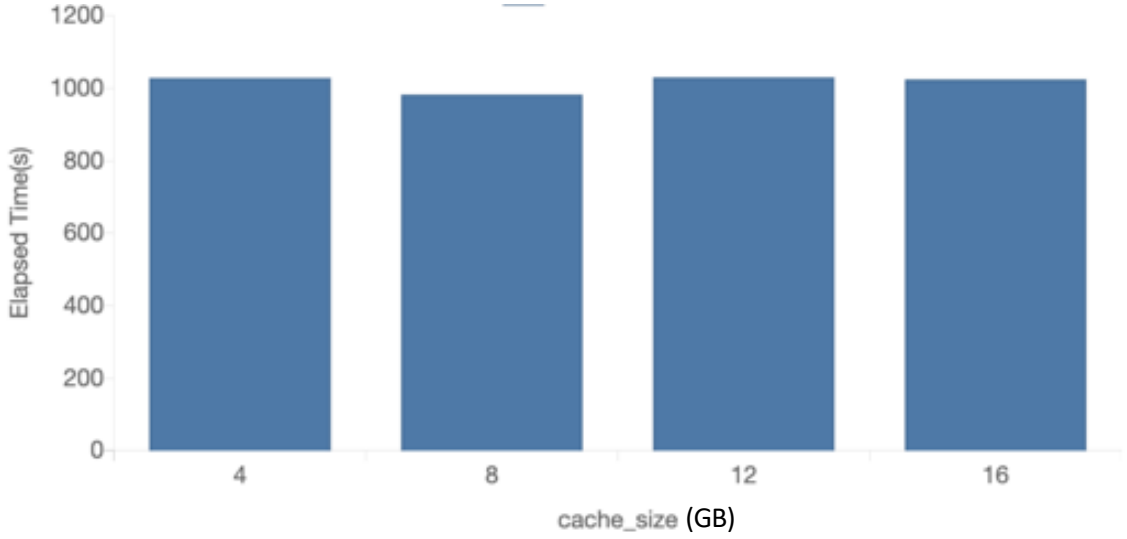


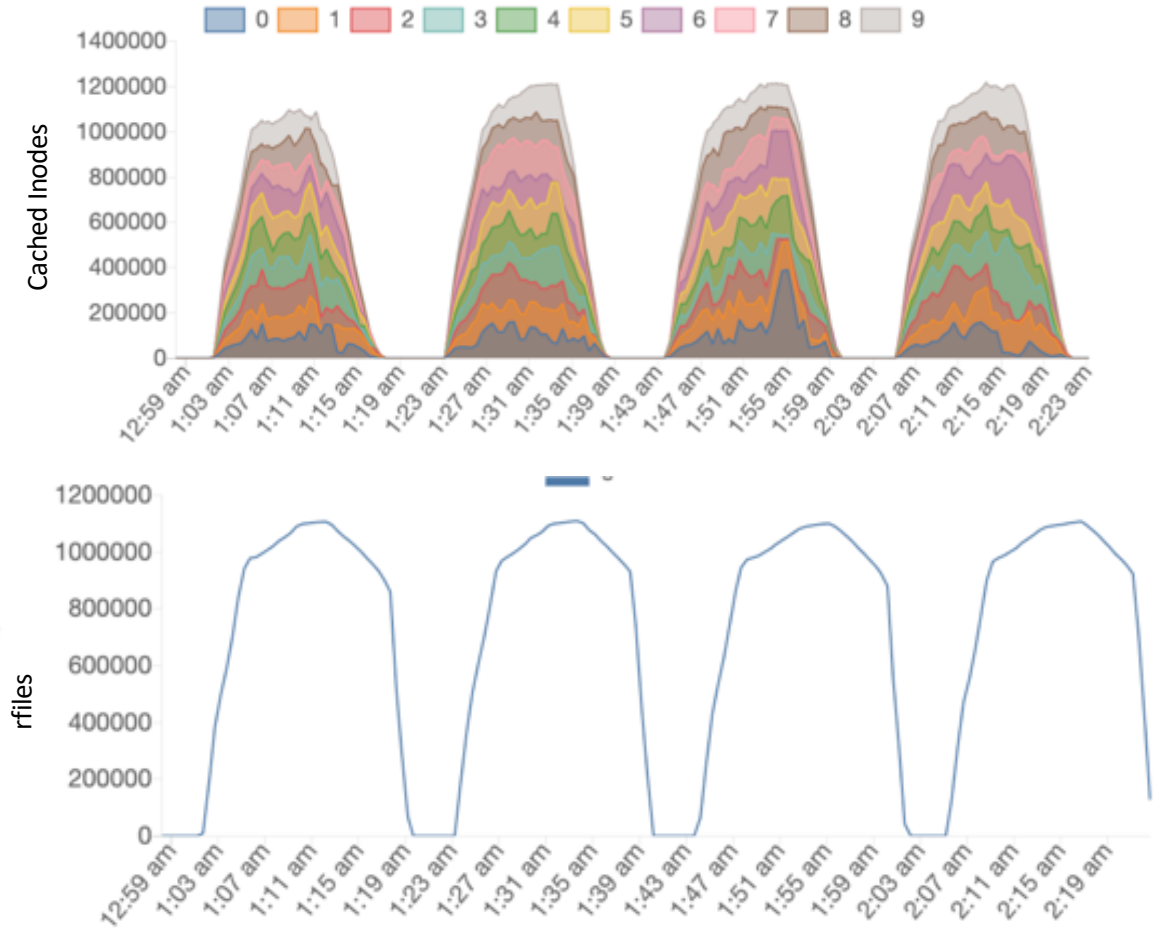




Impact of MDS Caches (kernel compile)

- Caches are enough to keep inodes in RAM

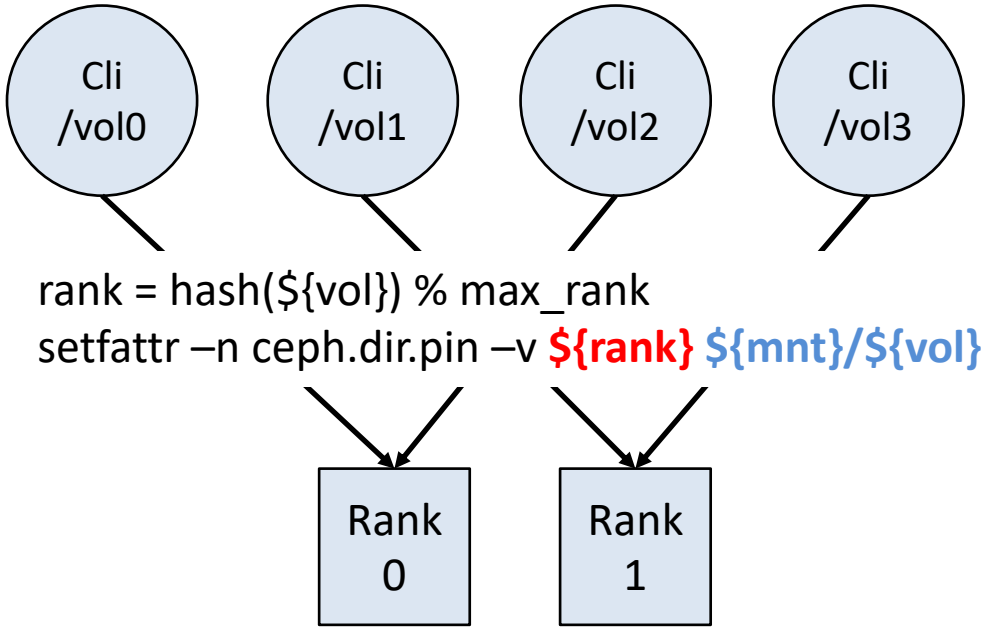




Test Configurations for Subtree Pinning

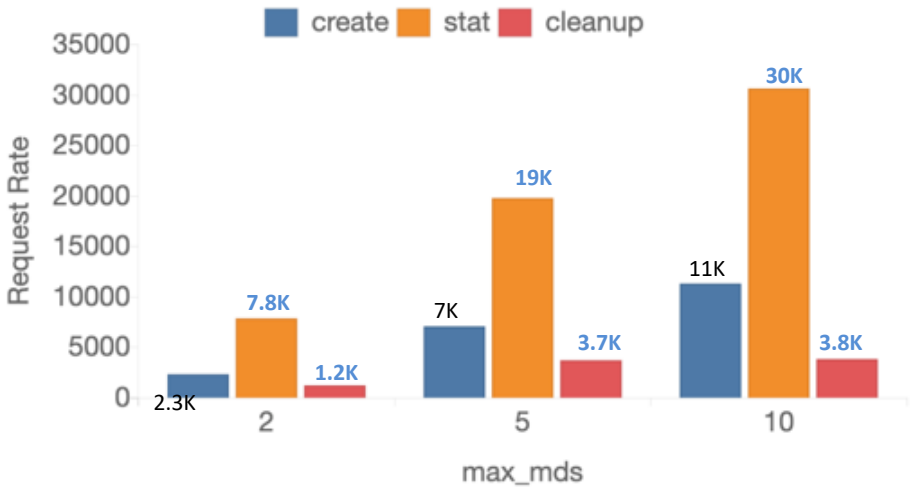
Type	Spec	
# of Active MDSs (max_mds)	2, 5, 10	
Per MDS Cache Size	1GB (mds_cache_memory_limit)	
# of Clients	20	
CephFS Client	Kernel cephfs driver	
Tool	Smallfile	Kernel compile
Files per Client	350K * max_mds / 20 (# of clients)	-
File Size	0	-
Files per Dir	17K (350K / 20)	-
Operation Type	Create,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

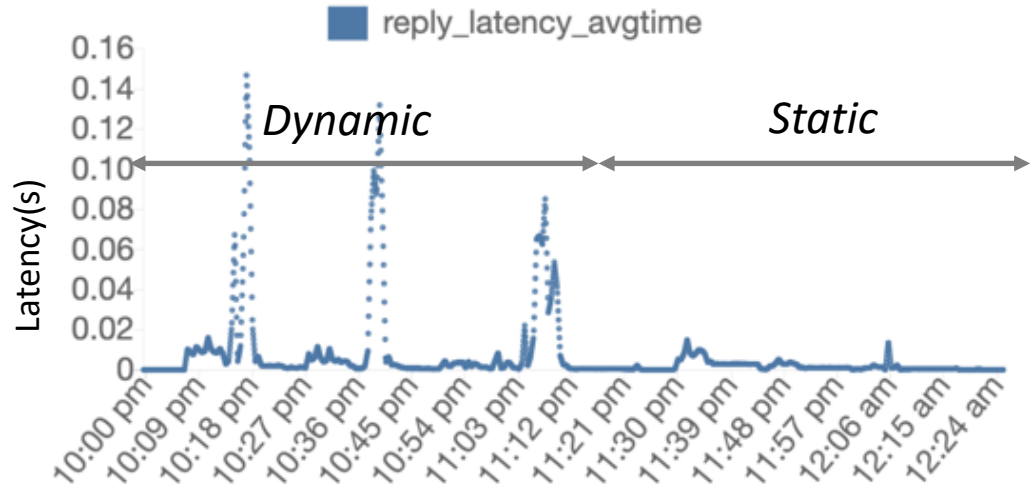
Example of Subtree Pinning



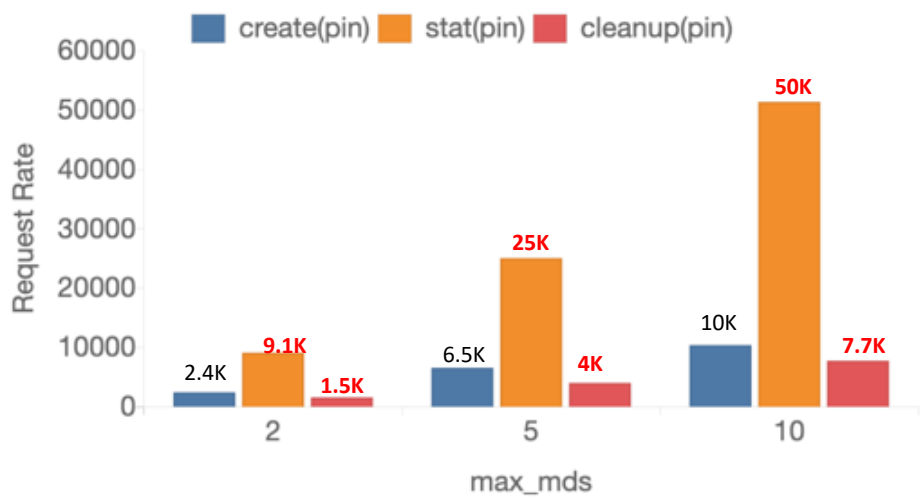
Subtree Pinning

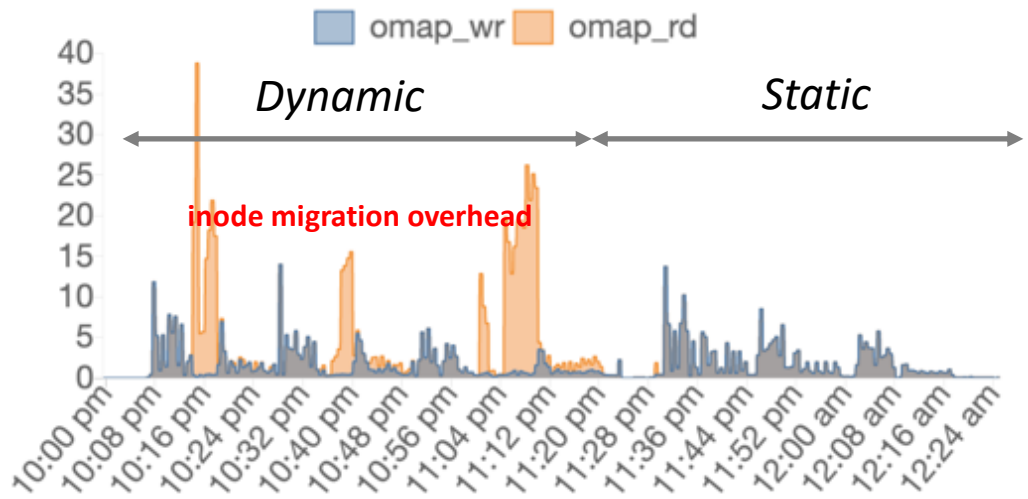
Dynamic Balancing (Default)





Static Pinning

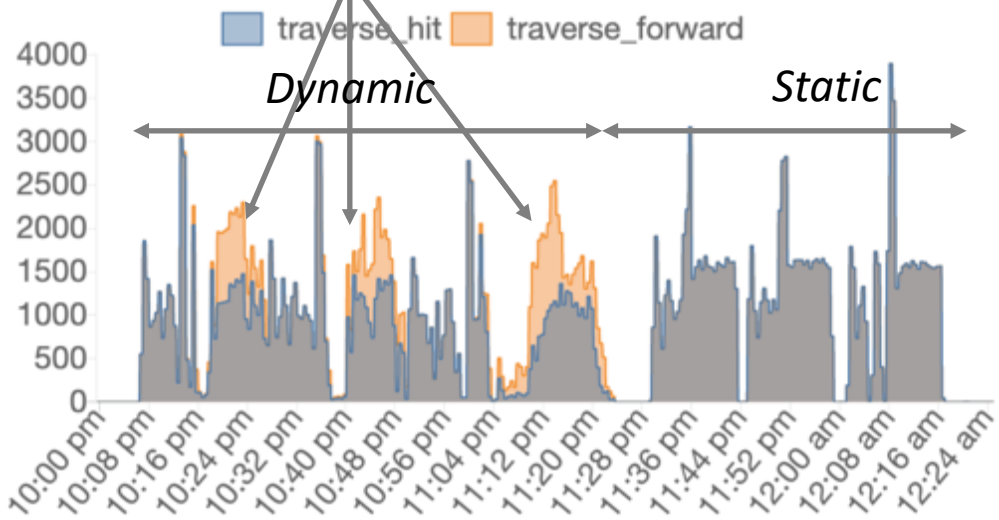




Dynamic balancing hurts internal lock and metadata update costs, leading to increased latencies

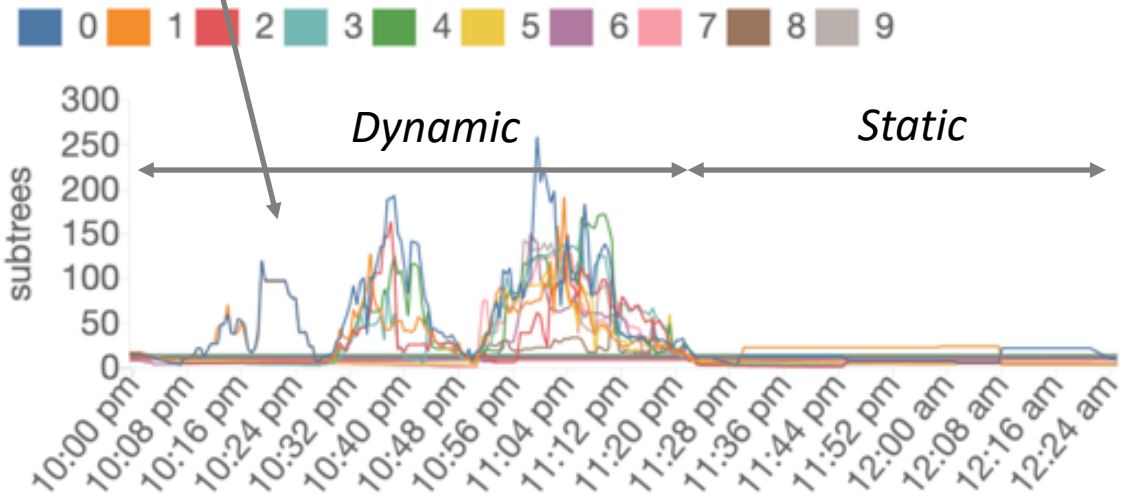
Subtree Pinning

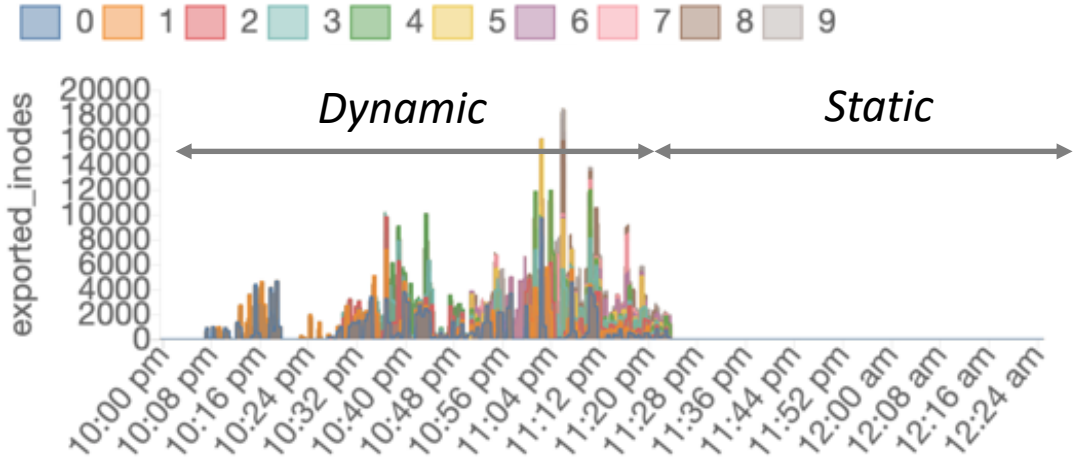
Forward traverse request to other ranks



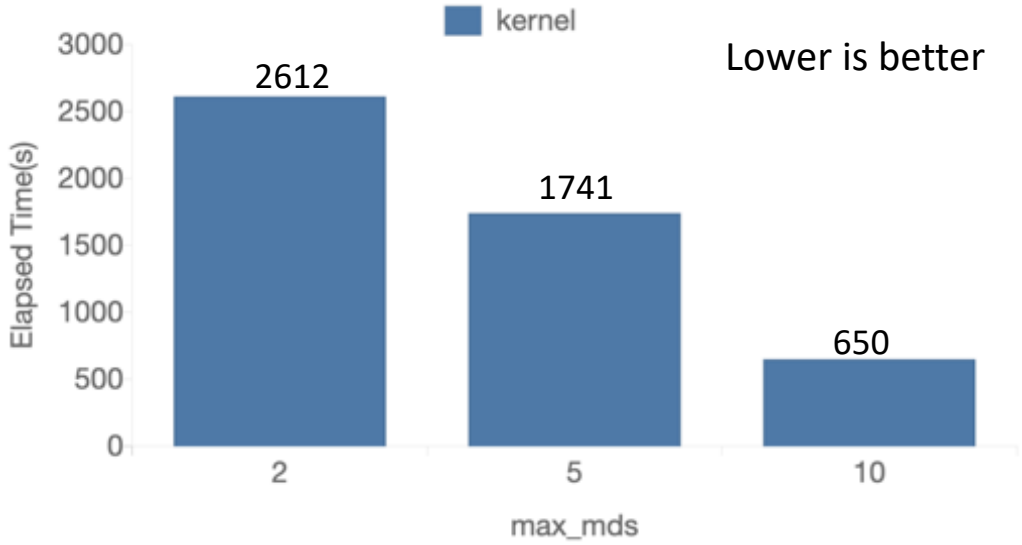
Rank0

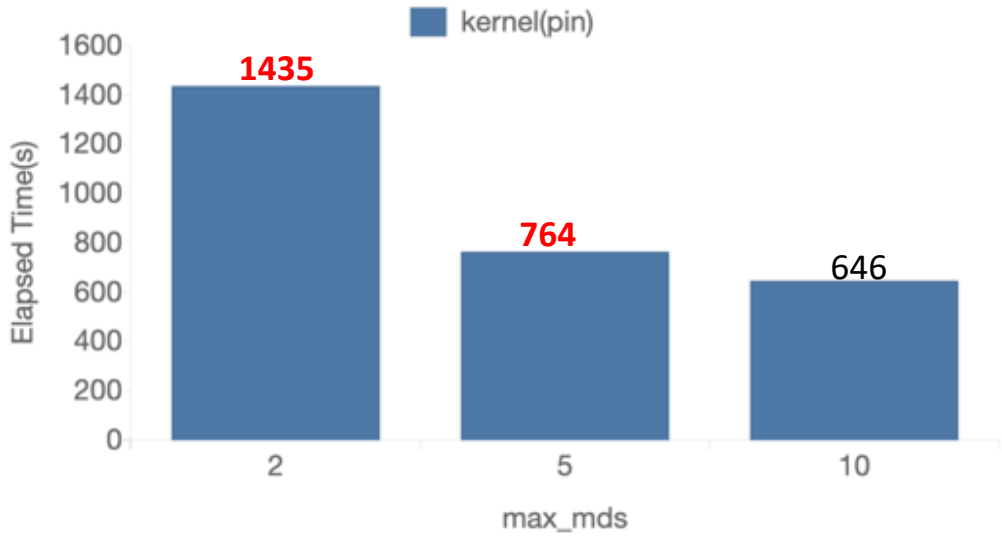
Increased metadata management cost (e.g., journaling)

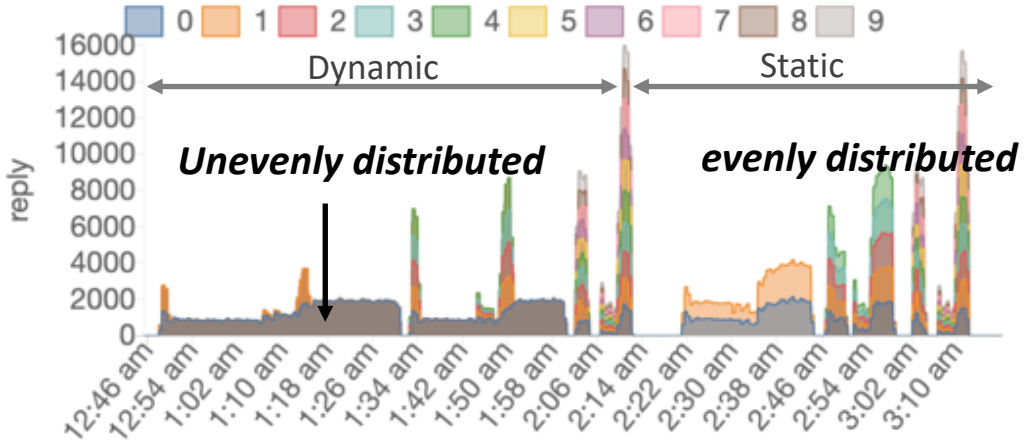


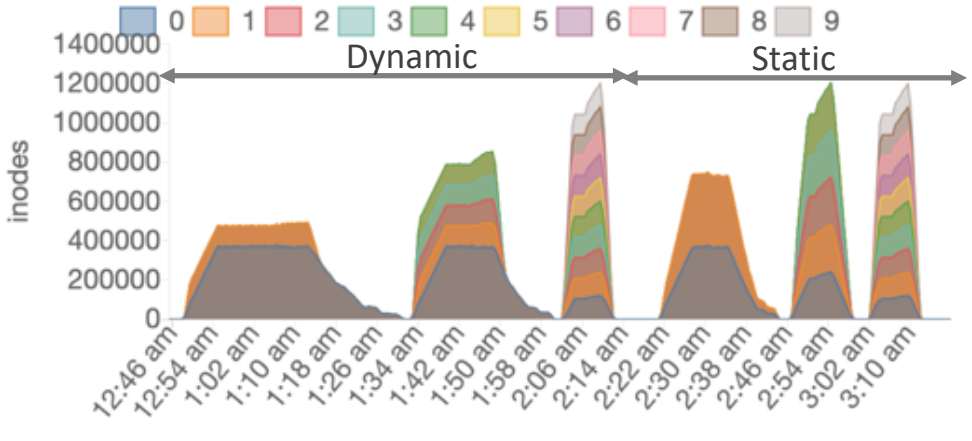


Subtree Pinning – Kernel Compile









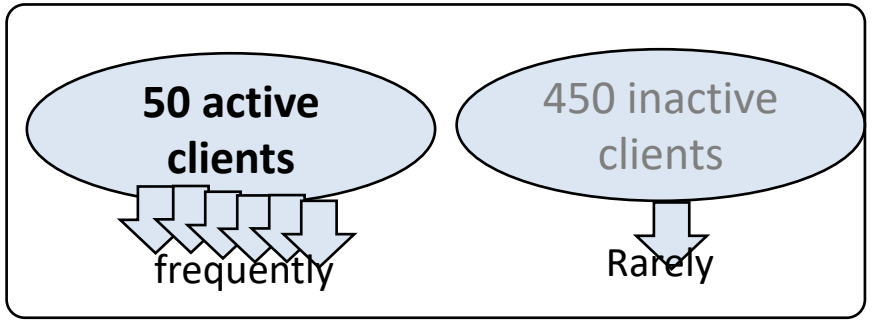
Subtree Pinning Pros/Cons

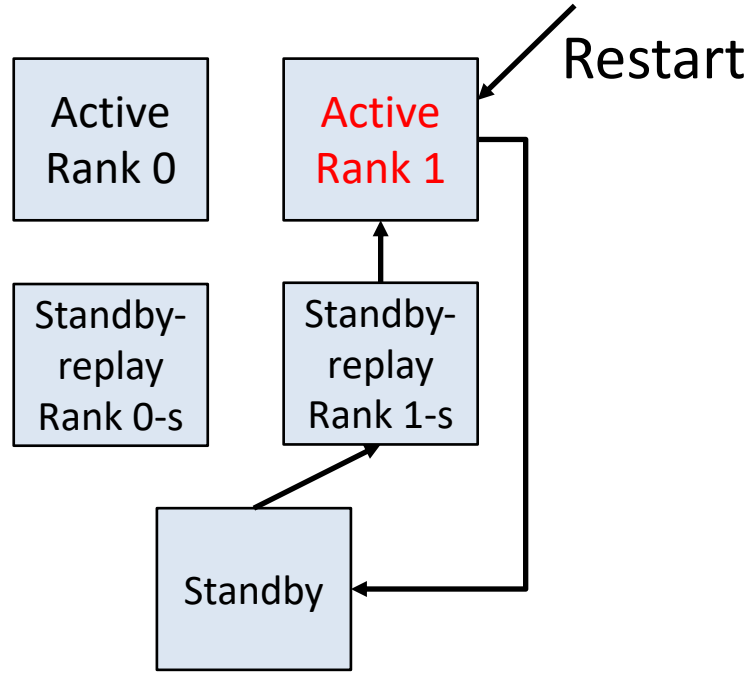
- Pros
  - Subtrees can clearly be isolated into specific MDS ranks
  - Improved cache locality results in the better performance
  - Exporting/importing subtrees among MDS ranks are prevented (e.g., minimizing balancing calculation and lock contention between MDS and clients)

- Cons
  - Load imbalance sometimes
  - Administrative burden → manually pinning (how?)
    - How do we know client workload dynamics?

Test Configuration for MDS Recovery

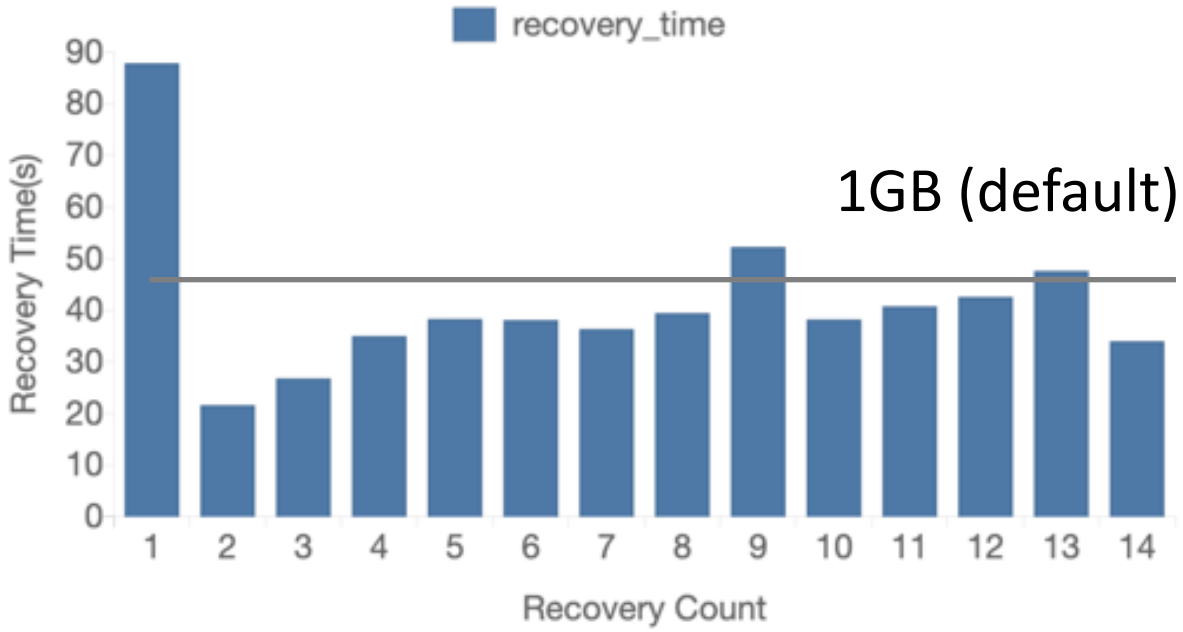
Type	Spec
# of Active MDSs (max_mds)	2
Standby-replay MDSs	2
Standby MDS	1
mds_cache_memory_limit	**1GB, 8GB, 16GB**
# of Clients	**50 active + (450 inactive)**
CephFS Client	Kernel cephfs driver
Tool	vdbench
width	10
files	Relies on mds_cache_memory_limit (mds_cache_memory_limit * 350K * max_mds / width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

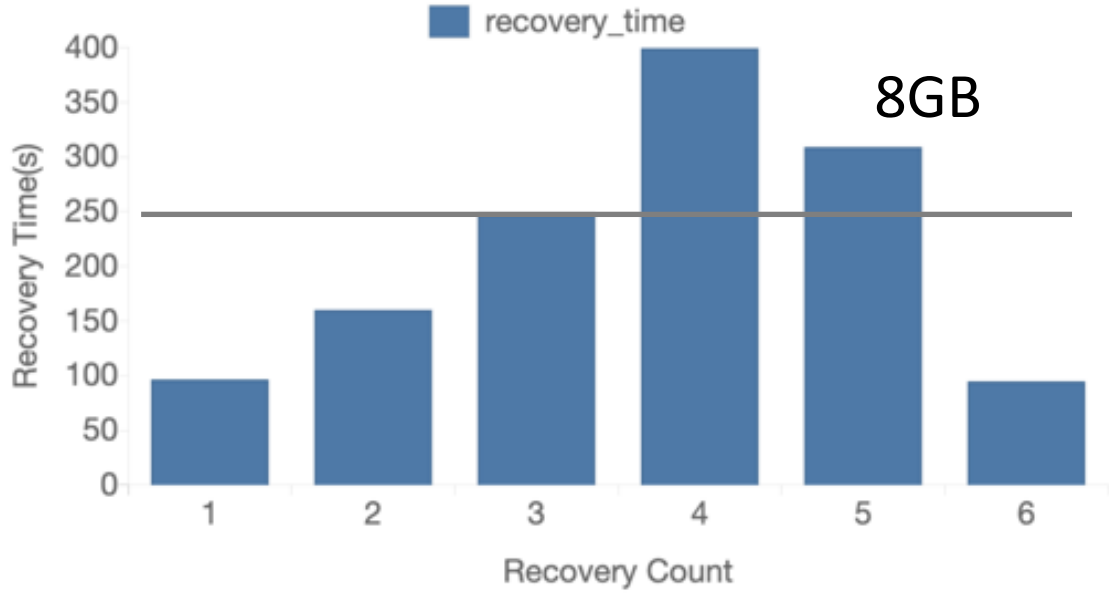


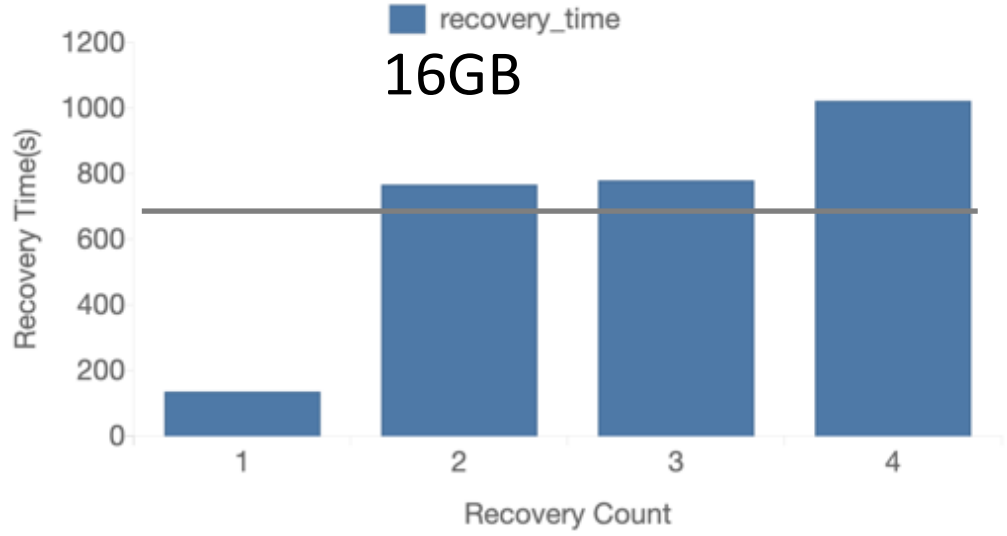


We measure the time when # of Active MDSs is changed from 1 to 2

Recovery Test Results

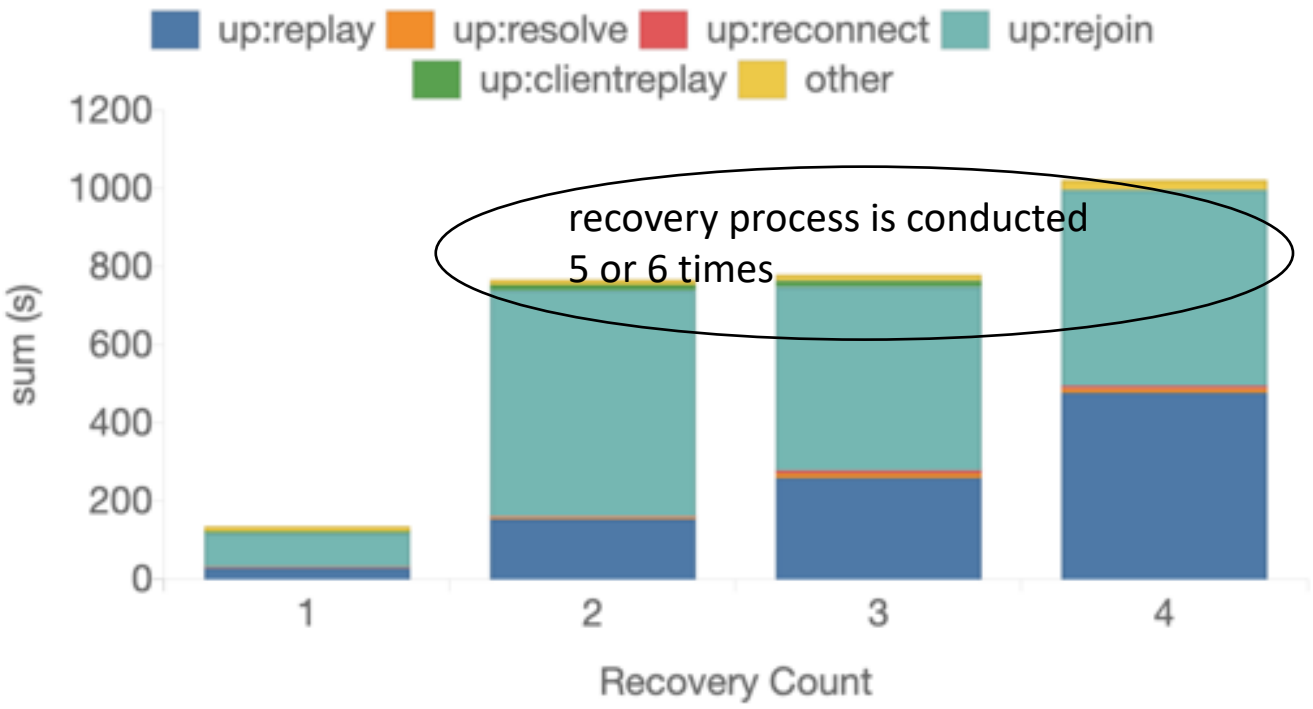






Recovery Time Breakdown for 16GB Cache

- “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

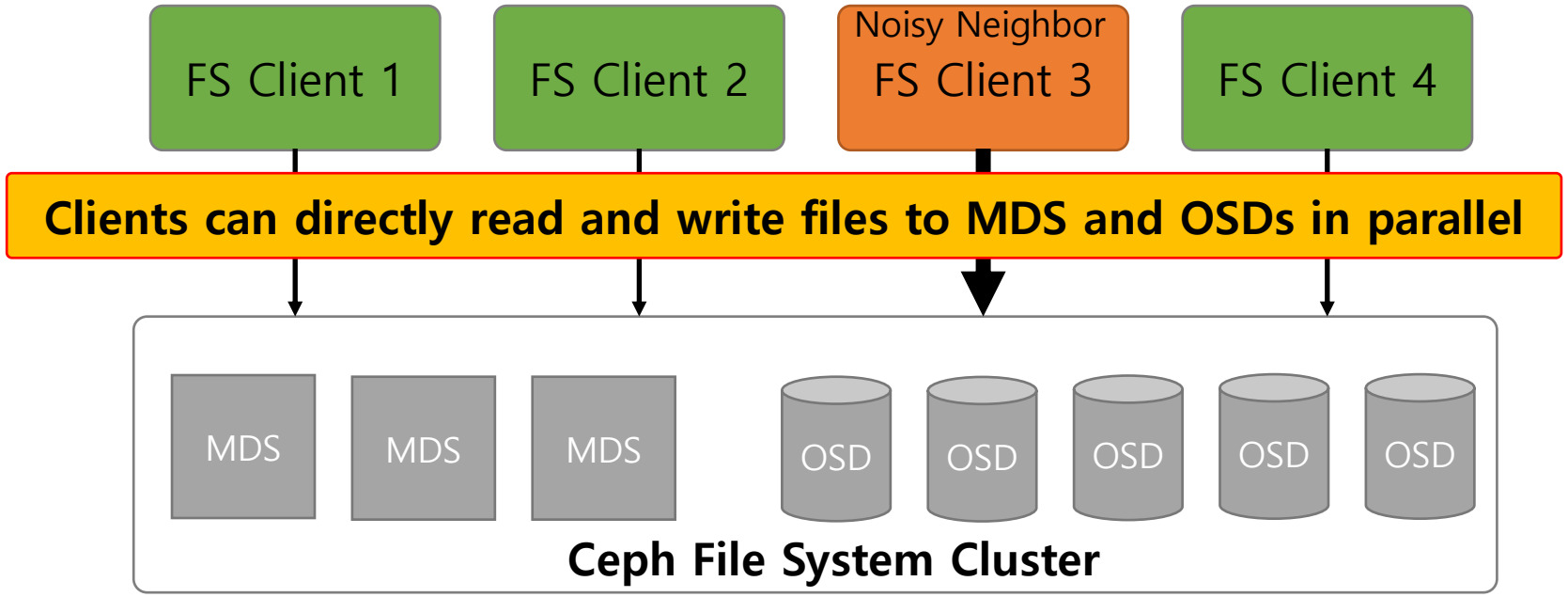
- CephFS Overview
- MDS Evaluation
  - MDS Scalability
  - CephFS Kernel vs FUSE clients
  - Impact of MDS Cache Size
  - Static Subtree Pinning
  - MDS Recovery Time
- **mClock QoS Scheduler for CephFS**
- Summary & Wrap Up

Storage Comparisons

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

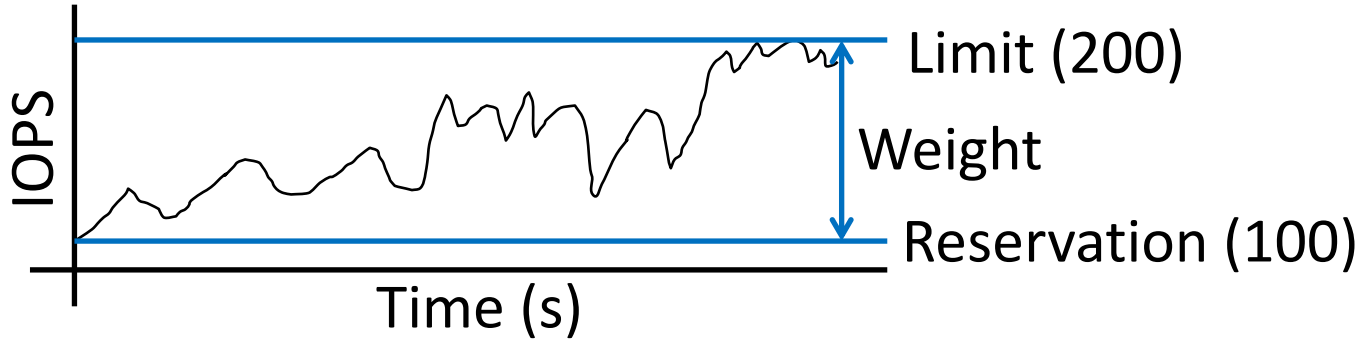
Technical Challenges

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



dmClock Scheduler Algorithm

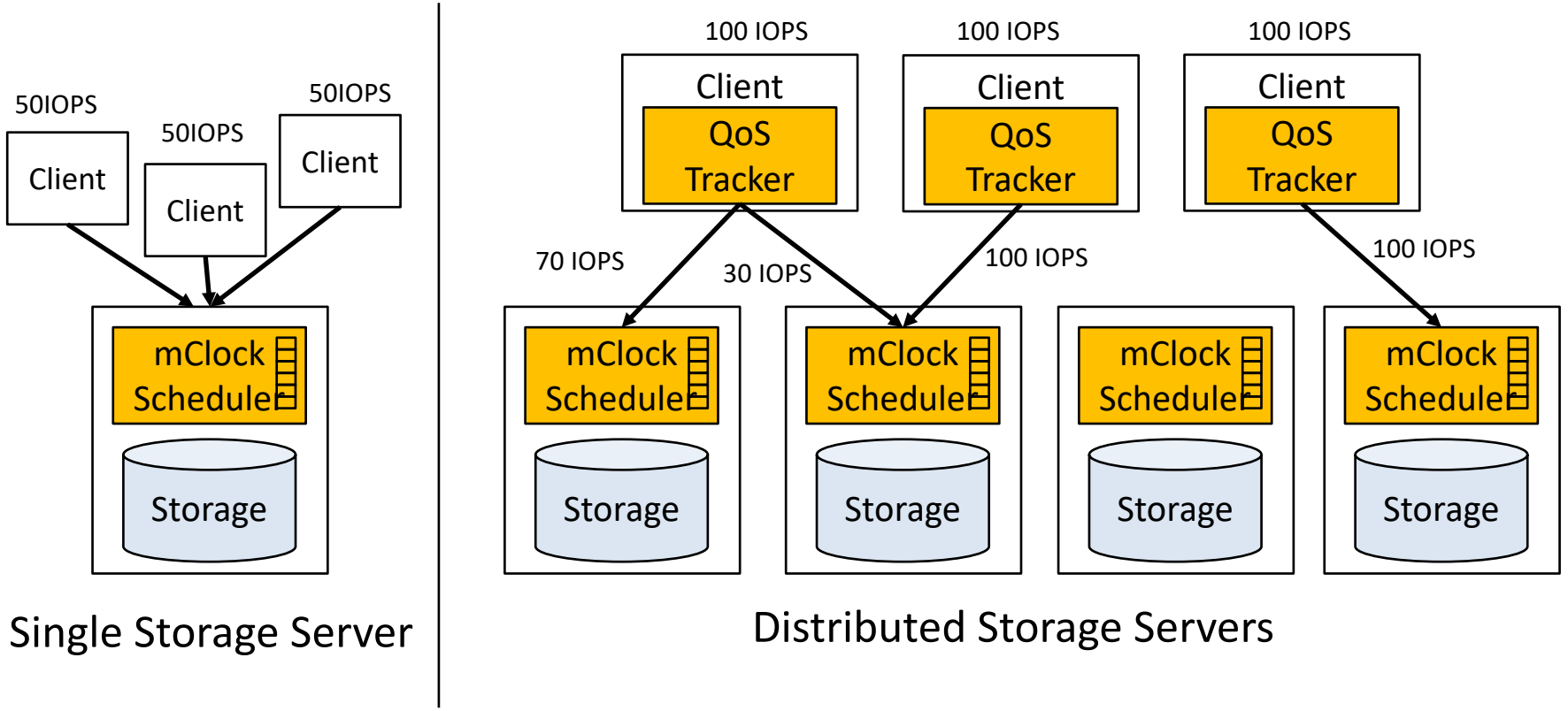
- The dmClock paper[1] has been published at USENIX OSDI'10 (developed by VMware)
- Three control knobs
  - **Reservation**: minimum guarantee
  - **Limit**: maximum allowed
  - **Weight**: proportional allocation
- Features of dmClock
  - Supports all controls in a single algorithm
  - Easy to implement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



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

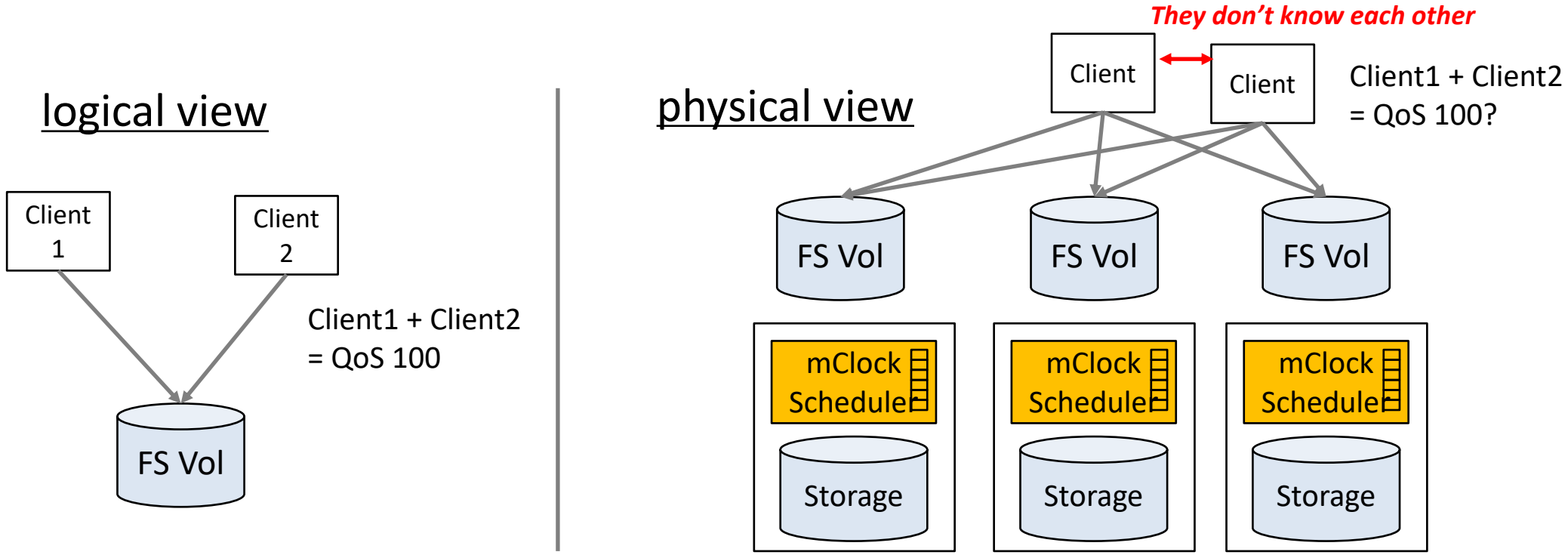
dmClock Scheduler Architecture

- 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

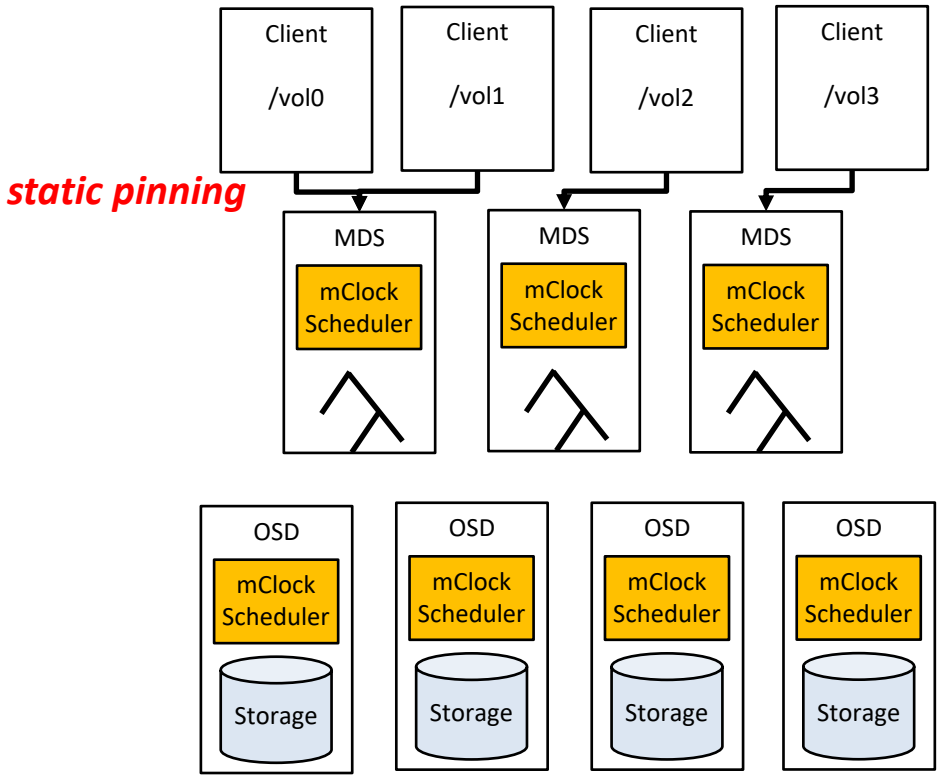
- **Modifications to the client SW stack**
  - Backward compatibility issue → what if the client disables the QoS feature
- **Multiple different clients on the same resource (e.g., volume)**
  - dmClock doesn't consider the volume share case



CephFS QoS Solutions

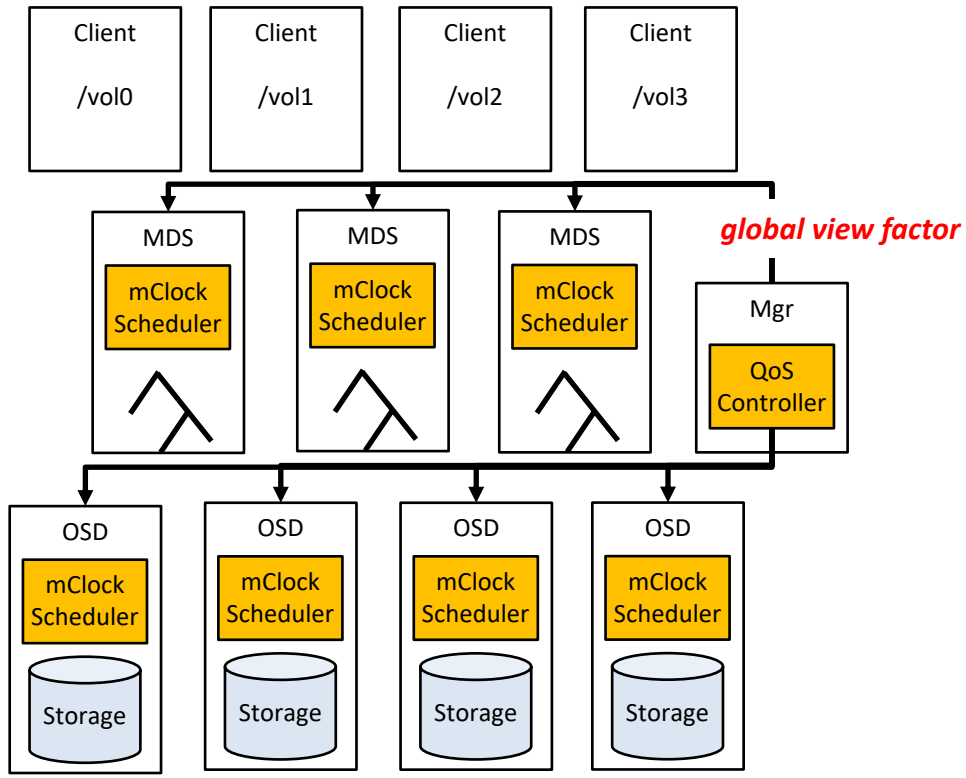
mClock + Static Pinning

Our Approach



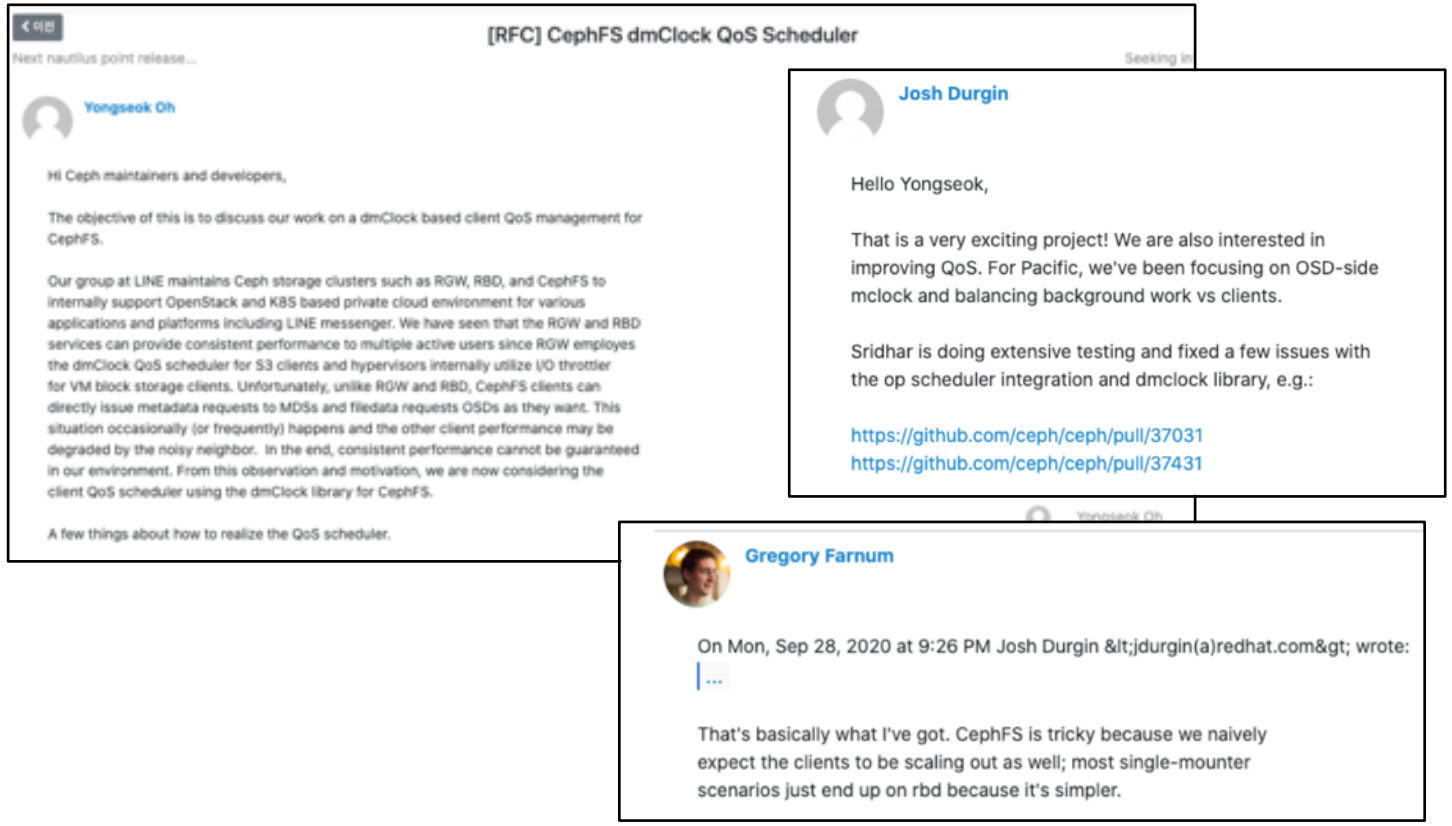
Global QoS Controller Solution

KAIST Research Project



Ceph Community Proposal

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



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

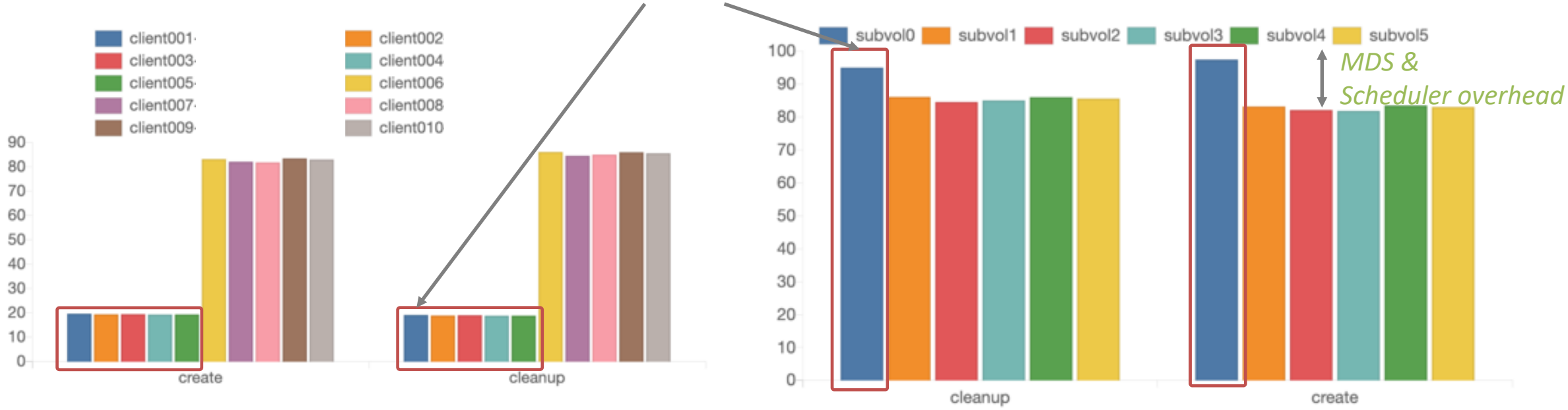
```
ceph-fuse -n client.admin -c ceph.conf -k keyring —client-mountpoint=/ mnt
setfattr -n ceph.dmclock.mds_reservation -v 1000 /mnt/volumes/_nogroup/subvolume/
setfattr -n ceph.dmclock.mds_weight -v 1000 /mnt/volumes/_nogroup/subvolume/
setfattr -n ceph.dmclock.mds_limit -v 1000 /mnt/volumes/_nogroup/subvolume/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

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/batrick/ceph-linode