A New Partitioning Scheme for CephFS

LINE
오용석

Ceph Days Korea 2023

Outline of Contents

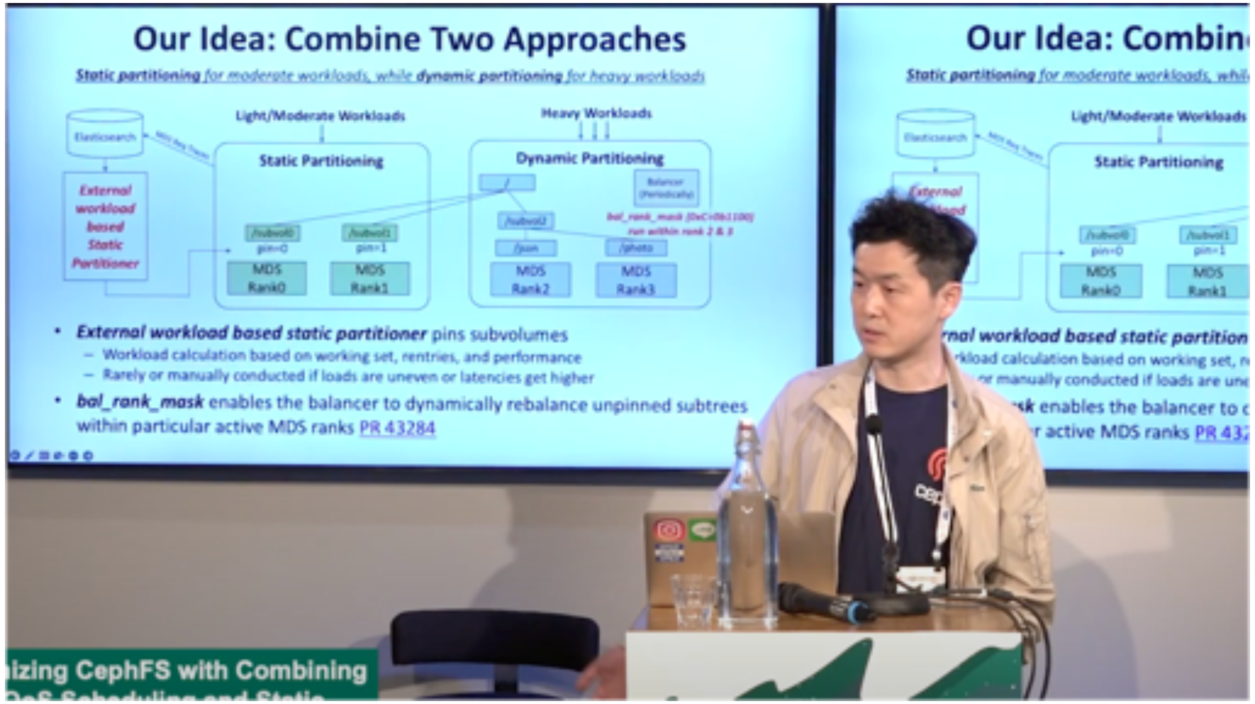
- Introduction and background
- CephFS subtree partitioning
- Combining static and dynamic partitioning schemes
  - Workload based static partitioner with bal_rank_mask
  - A new CephFS MDS balancer with ceph.dir.bal.mask
- Future works and conclusions

Who Am I?

- Yongseok Oh (오용석)
- 2015 Ph.D. in Computer Science
  - Flash based storage systems
  - Research papers in FAST, MSST, Systor (over 400 citations)
- 2015 SKT - SW-defined Storage Lab.
  - Enterprise NVMe SSD FW
  - SPDK based file system
- 2017 SK hynix - FW Group
  - Enterprise NVMe SSD FW
  - ZNS, global wear leveling, SR-IOV, NVMe-MI
- 2020 ~ LINE Plus - Cloud Storage
  - CephFS based shared file service
  - NVMeoF based storage service
- Hobby: camping, boxing, jogging

Cephalocon2023

- **Optimizing CephFS with Combining MDS QoS Scheduling and Static-Dynamic Subtree Partitioning**



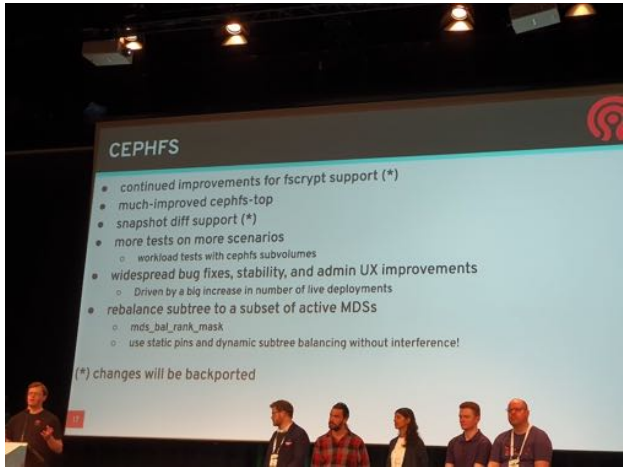
Our Idea: Combine Two Approaches

Static partitioning for moderate workloads, while dynamic partitioning for heavy workloads

- External workload based static partitioner pins subvolumes
  - Workload calculation based on working set, rentries, and performance
  - Rarely or manually conducted if loads are uneven or latencies get higher
- bal_rank_mask enables the balancer to dynamically rebalance unpinned subtrees within particular active MDS ranks PR 43284

https://www.youtube.com/watch?v=pDURlI6Y-Ug

Cephalocon2023


CEPHFS

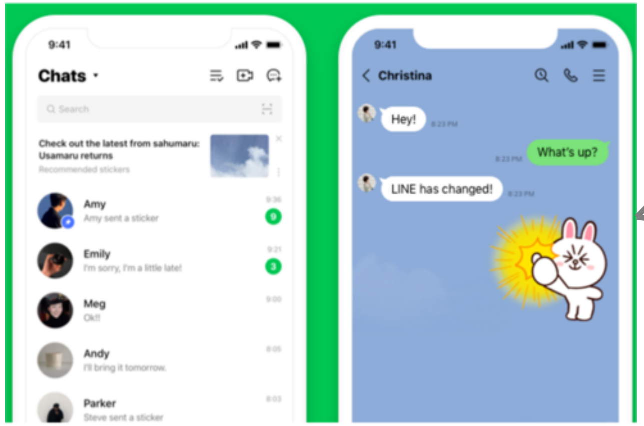
- continued improvements for fscrypt support (*)
- much-improved cephfs-top
- snapshot diff support (*)
- more tests on more scenarios
  - workload tests with cephfs subvolumes
- widespread bug fixes, stability, and admin UX improvements
  - Driven by a big increase in number of live deployments
- rebalance subtree to a subset of active MDSs
  - mds_bal_rank_mask
  - use static pins and dynamic subtree balancing without interference!

(*) changes will be backported

17

LINE

- LINE, a messaging and communication service with over **150 million** daily active users
- OpenStack and Kubernetes as a cloud Infrastructure
- Ceph as a software defined storage
  - RBD, RGW, and then **CephFS**



9:41

Chats

Search

Check out the latest from sahumaru: Usamaru returns

Recommended stickers

Amy
Amy sent a sticker

Emily
I'm sorry, I'm a little late!

Meg
Ok!!

Andy
I'll bring it tomorrow.

Parker
Steve sent a sticker

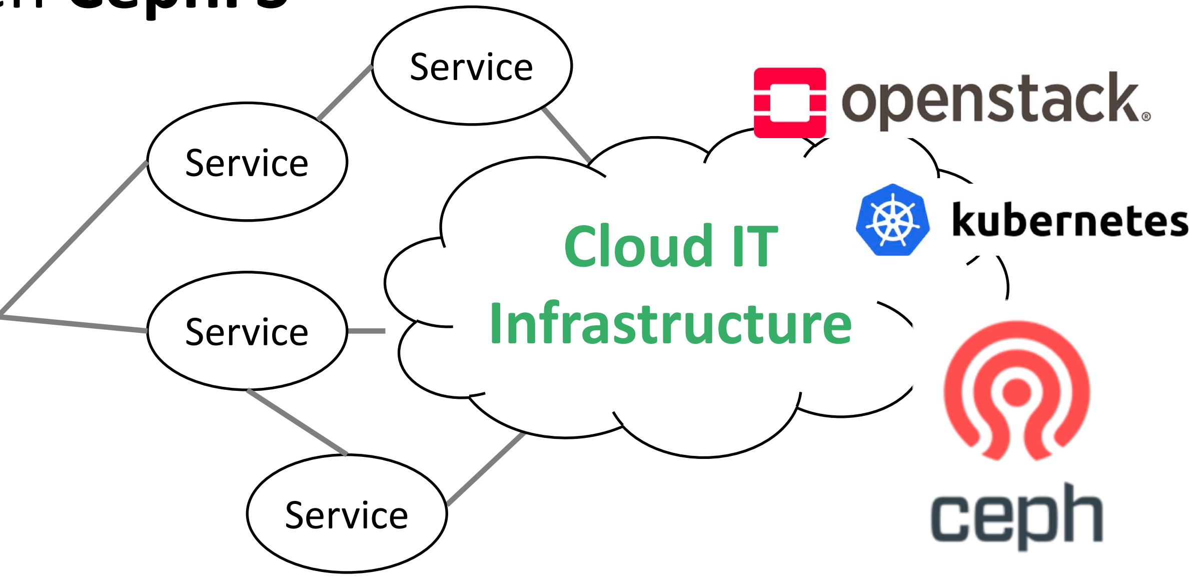
9:41

Christina

Hey!

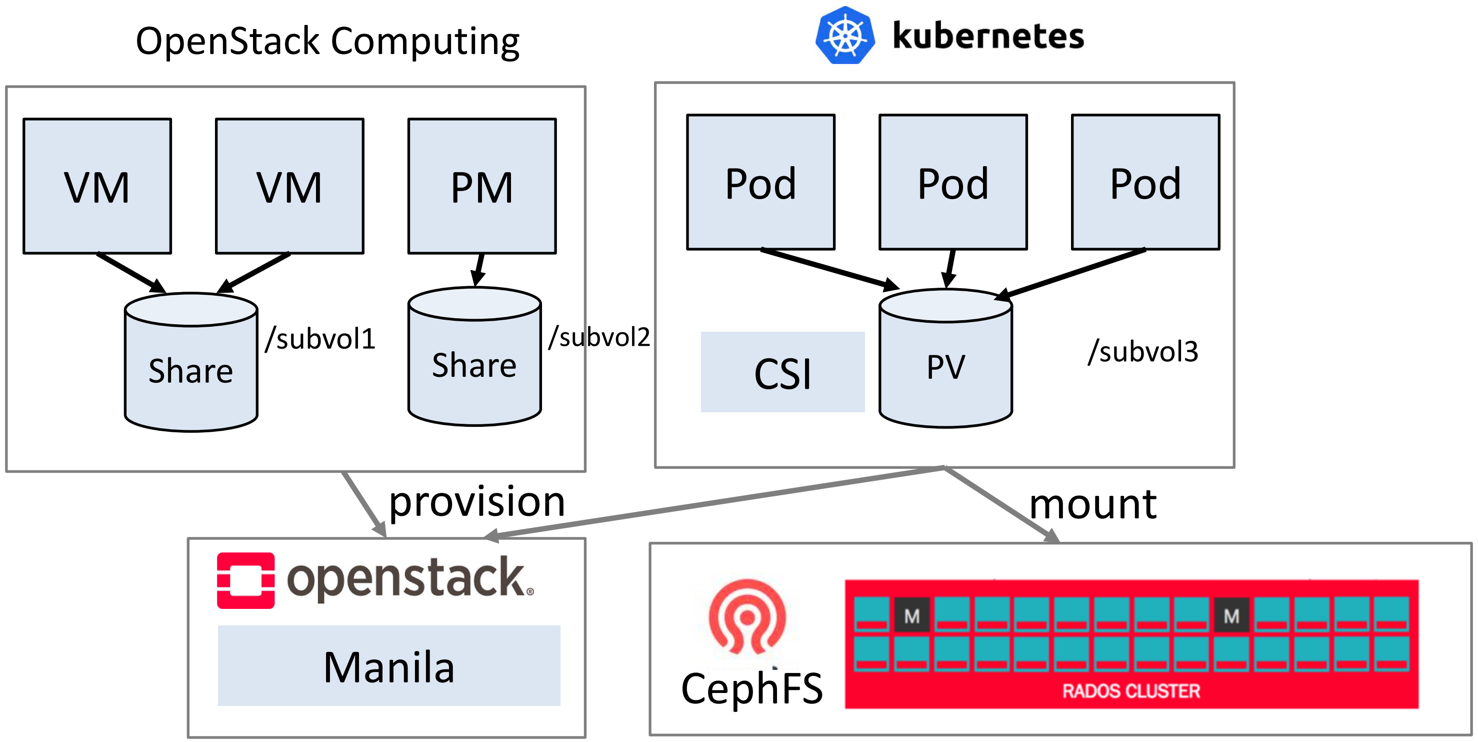
What's up?

LINE has changed!



Use Cases of CephFS in LINE

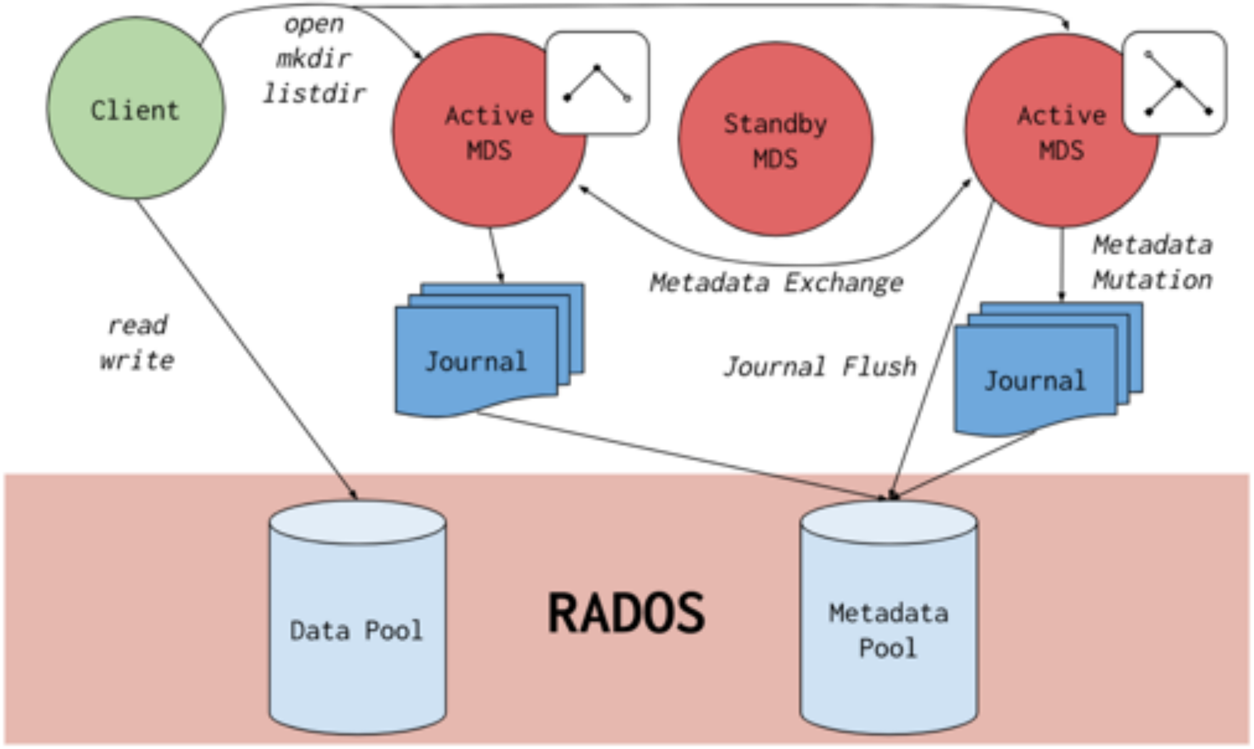
- CephFS as a shared file system for clouds
  - OpenStack Manila to manage subvolumes of CephFS
- File system can be shared to VMs/Pods (major benefit)
  - NAS, ML training, speech data, backup

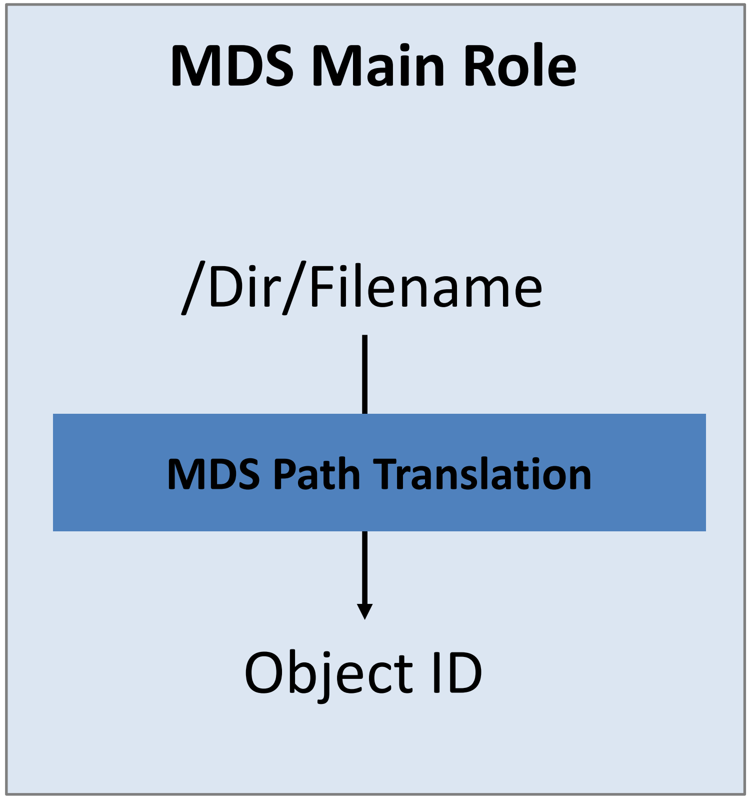


Key Features of CephFS

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

CephFS Architecture





MDS Main Role

/Dir/Filename

MDS Path Translation

Object ID

CephFS Clusters in LINE

- Major two clusters of our environment

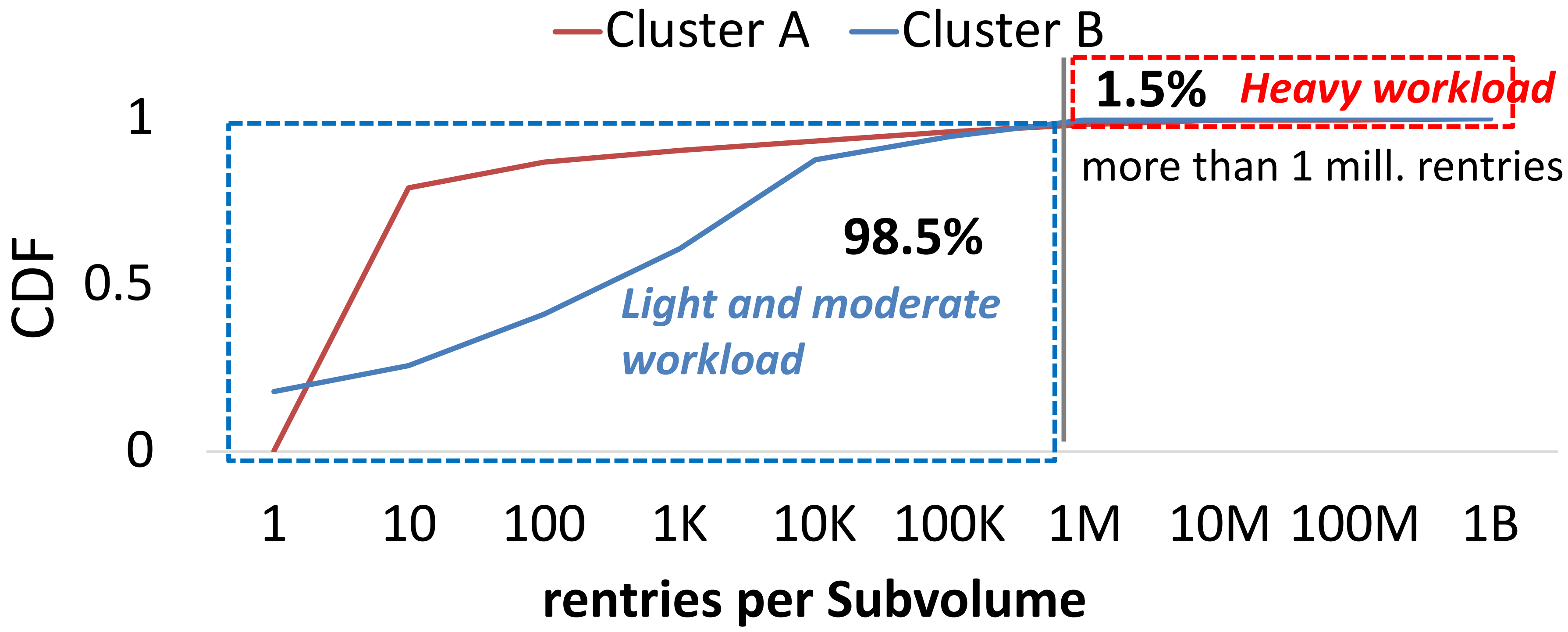
	Cluster A	Cluster B
Workload	General Purpose	ML, Data Processing
Active MDSs (on PMs + VMs)	30	36
OSDs (SSD based)	PB scale	PB scale
Sessions	> 5,000	> 5,000
Subvolumes	> 3,000	> 3,000

On Deploying CephFS in Production

- Sep 2020: CephFS service with two active MDSs + standby
- Jan 2021: technical issues and outages with growing users
  - Noisy neighbor, MDS slow request by balancer, increased recovery time
- Feb 2021: simple static partitioning has been applied
- Sep 2021: proposed MDS QoS scheduler (e.g., PR 38506)
- Dec 2021: enhanced MDS recovery time (e.g., PR 44246 , PR 41358)
  - Becoming up:active took 2 hours because of heartbeat timed out
- Aug 2022: the number of MDSs has increased to up to 36

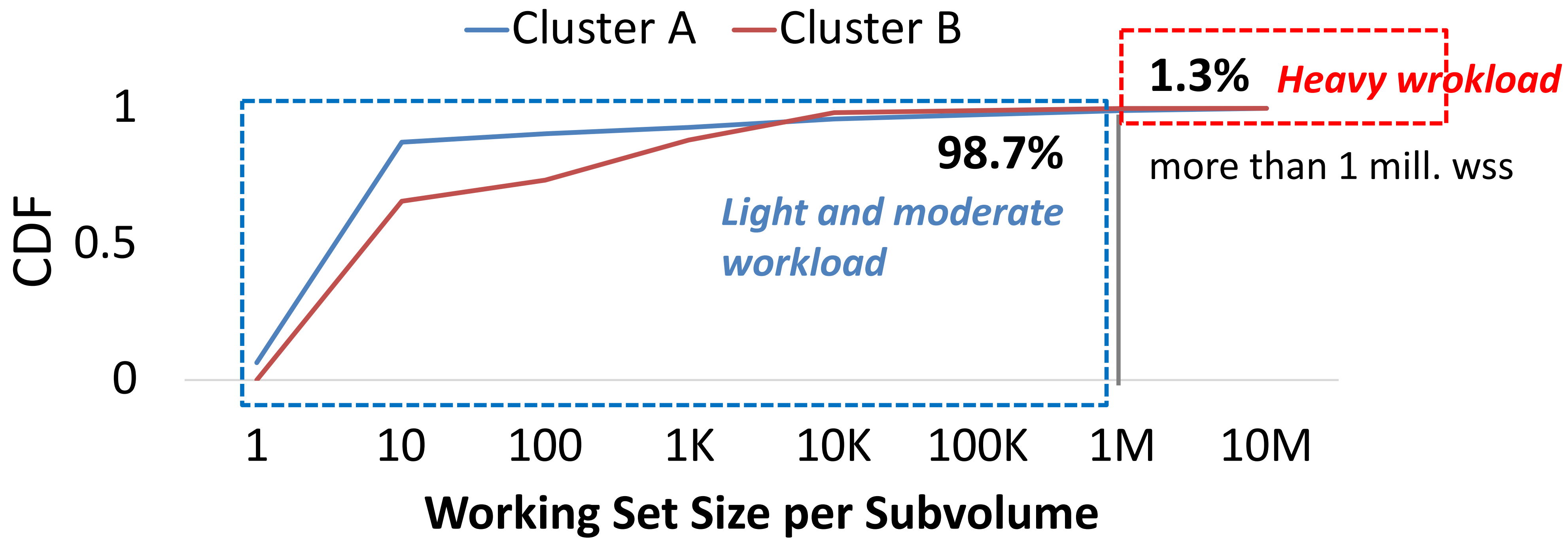
Now, we are working on ***subtree partitioning!***

MDS Workloads: rentries Distribution



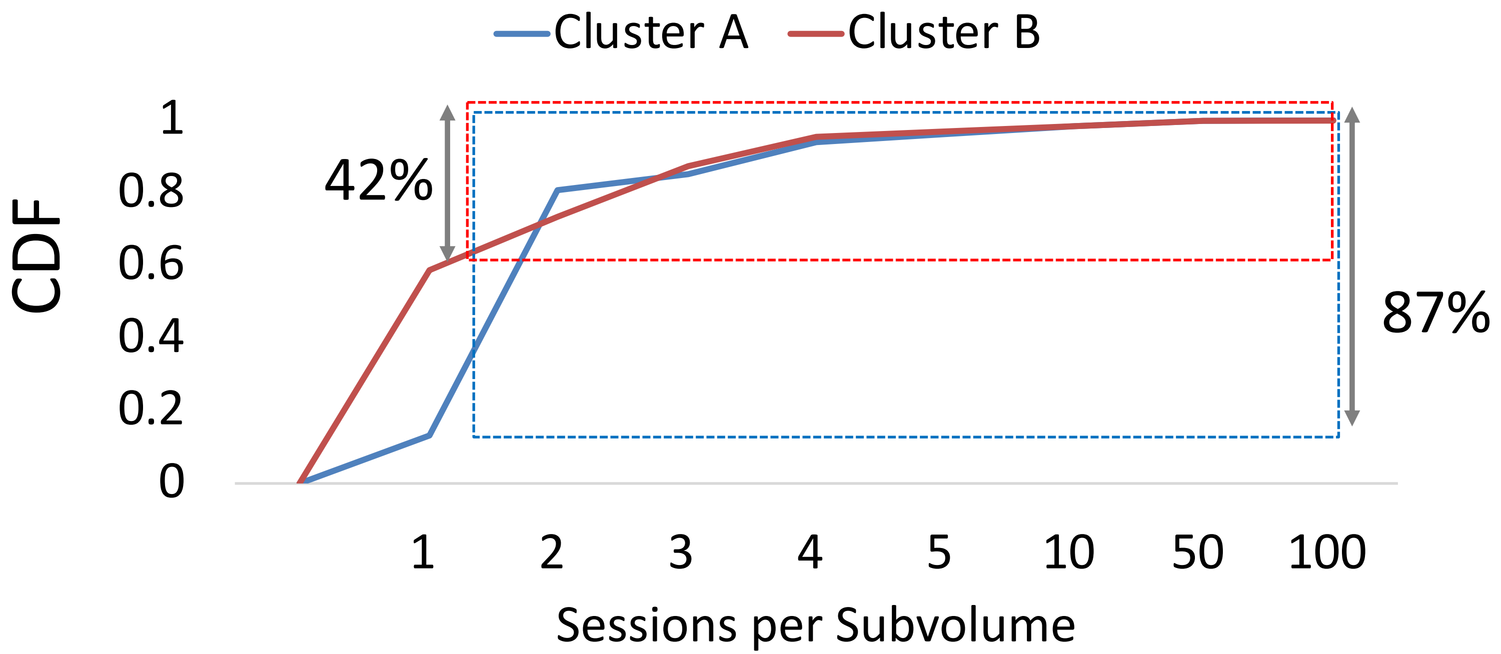
- **rentries** (e.g., files + dirs) per subvolume (measured via ceph.dir.rentries)
  - 98.5% of subvolumes contain rentries less than or equal to 1 mil.
  - Remaining of subvolumes have more than 1 mil rentries
- **Heavy workload** may affect the cache hit ratio, leading to performance degradation

MDS Workloads: Working Set Distribution



- **Working set size** (WSS) per subvolume (collected via uniq() of Elasticsearch)
  - 98.7% subvolumes have WSS of less than 1 mil.
- Heavy workload requires the large cache memory

MDS Workloads: Sessions per Subvolume



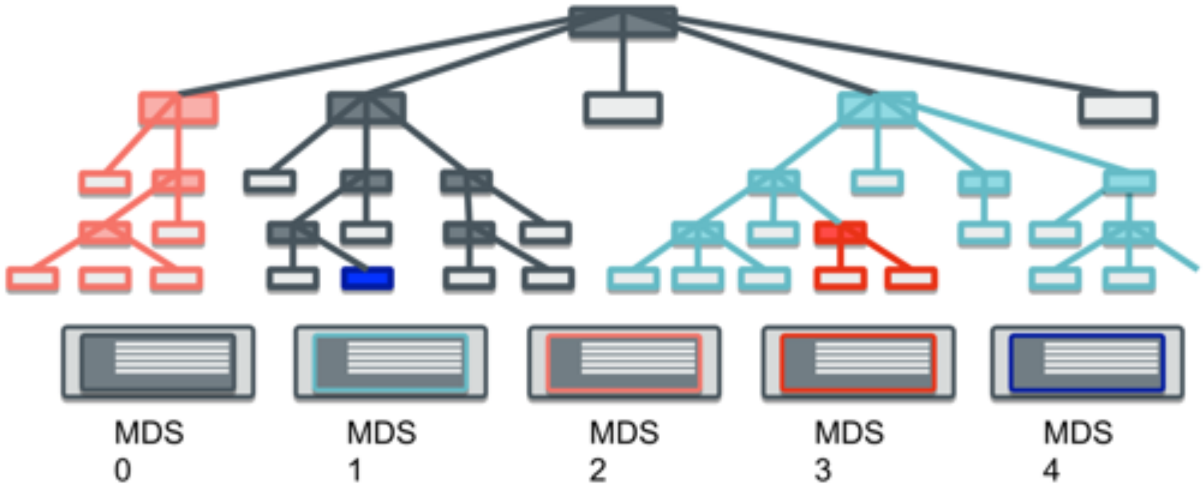
- 42% and 87% subvolumes are shared with sessions that have at least two sessions
  - Many sessions have the potential to generate heavy workloads

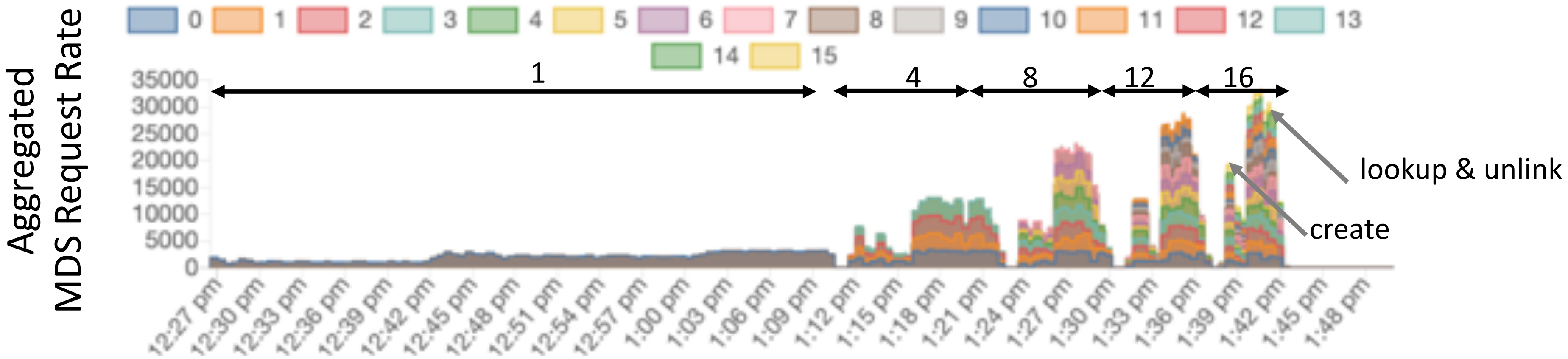
Outline of Contents

- Introduction and background
- CephFS subtree partitioning
- Combining static and dynamic partitioning schemes
  - Workload based static partitioner with bal_rank_mask
  - A new CephFS MDS balancer with ceph.dir.bal.mask
- Future works and conclusions

MDS Scalability with Subtree Partitioning

Can MDS performance scale with multiple MDSs?

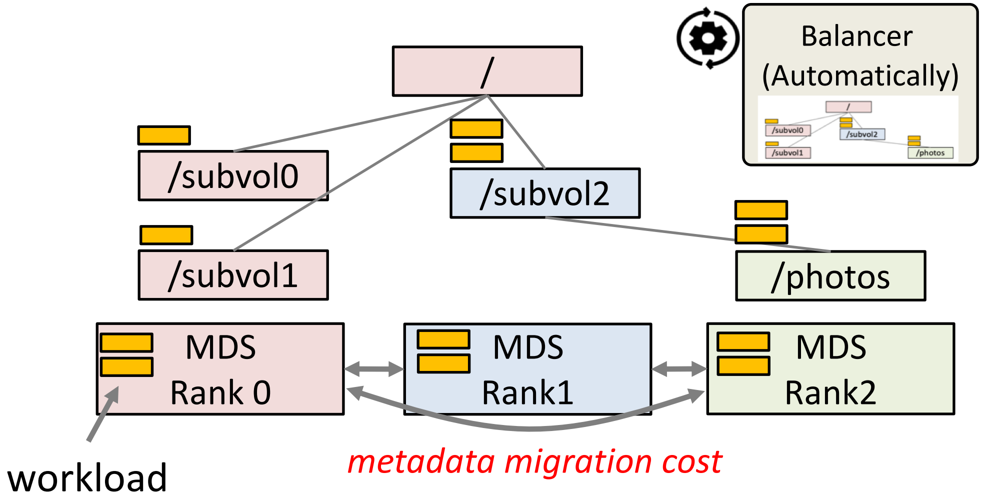




Dynamic vs Static Partitioning

Dynamic Subtree Partitioning

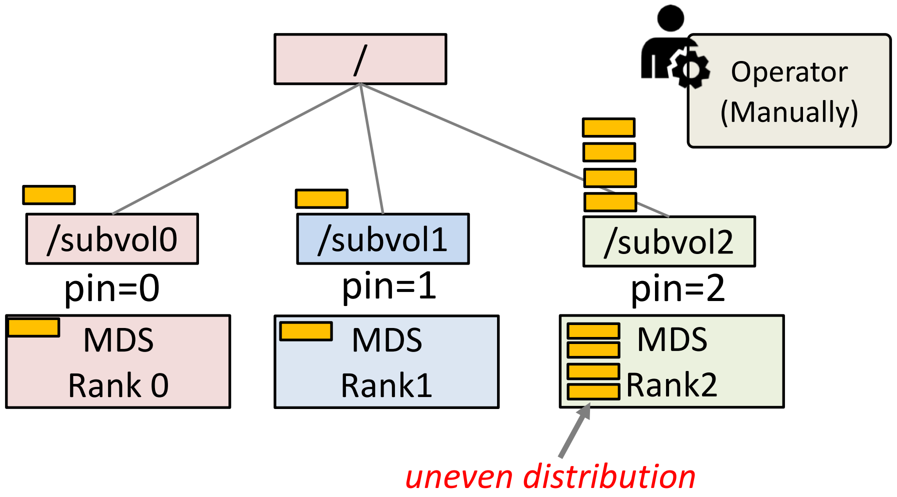
Periodically redistribute subtrees



metadata migration cost

Static Subtree Partitioning

Simply pin subtrees to their own ranks



uneven distribution

Pros

- MDS horizontal scalability

Cons

- Metadata migration cost

Pros

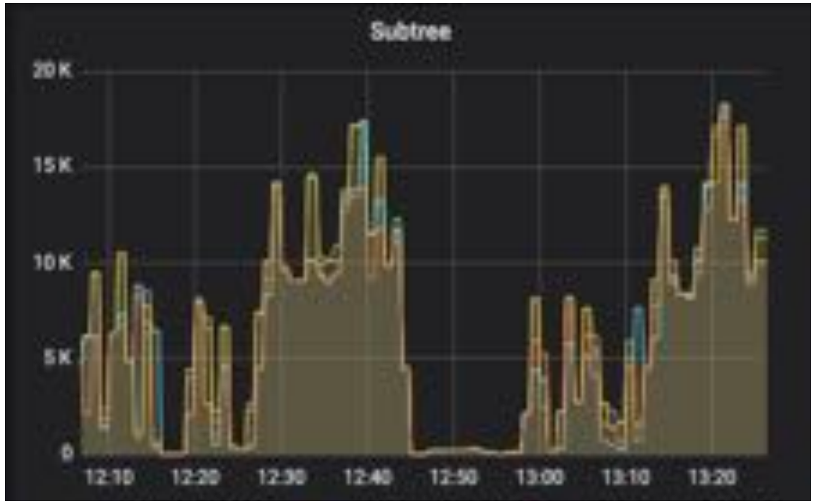
- Negligible migration cost
- Provisioned performance

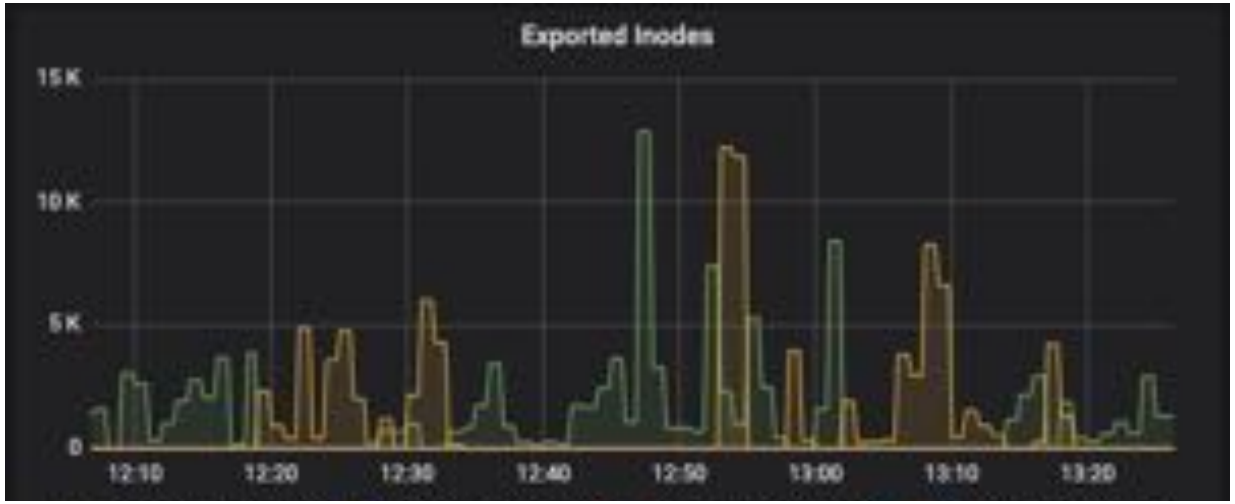
Cons

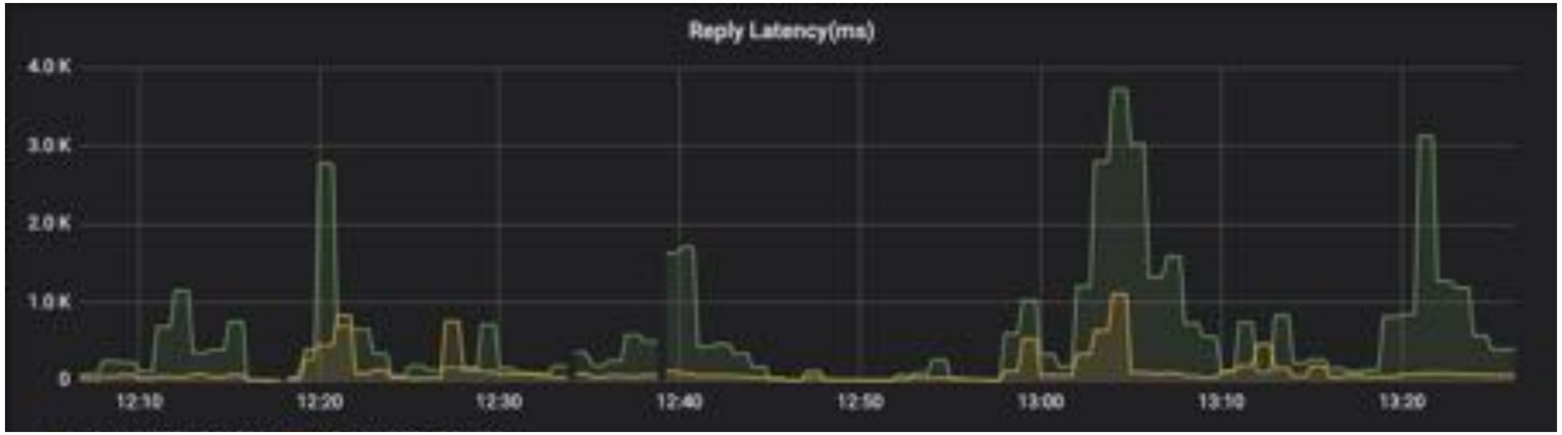
- Uneven workload distribution
- Additional operator efforts

Negative Impact of Dynamic Partitioning

- Increased subtree changes and inode migrations







Migrating Subtrees Incurs MDS Slow Requests

```
"MDS_SLOW_REQUEST": {
        "severity": "HEALTH_WARN",
        "summary": {
          "message": "2 MDSs report slow requests"
        },
        "detail": [
          {
            "message": "mds(mds.1): 1253 slow requests are blocked > 30 secs"
          },
          {
            "message": "mds(mds.0): 1 slow requests are blocked > 30 secs"
          }
        ]
}
```

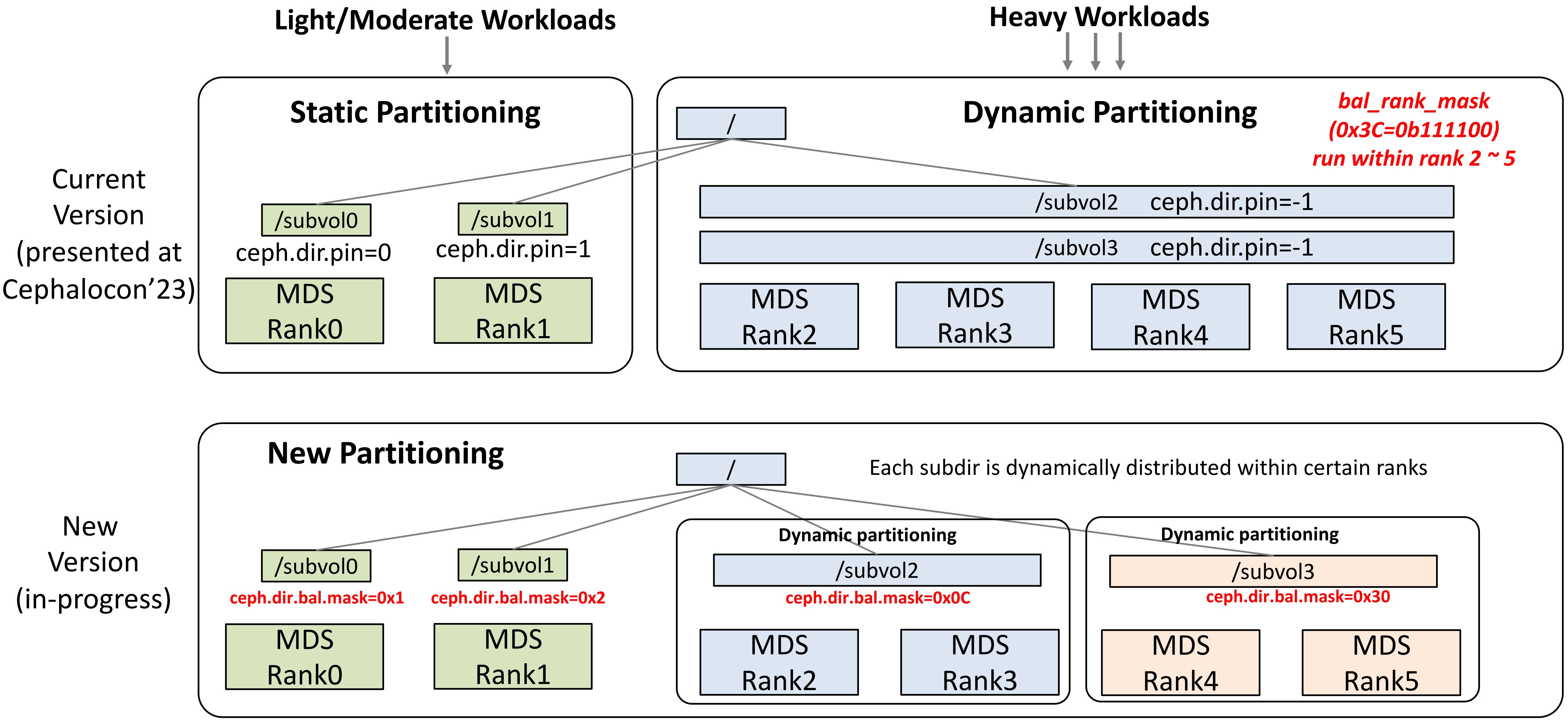
We have observed slow requests are resolved since employing static partitioning!

Outline of Contents

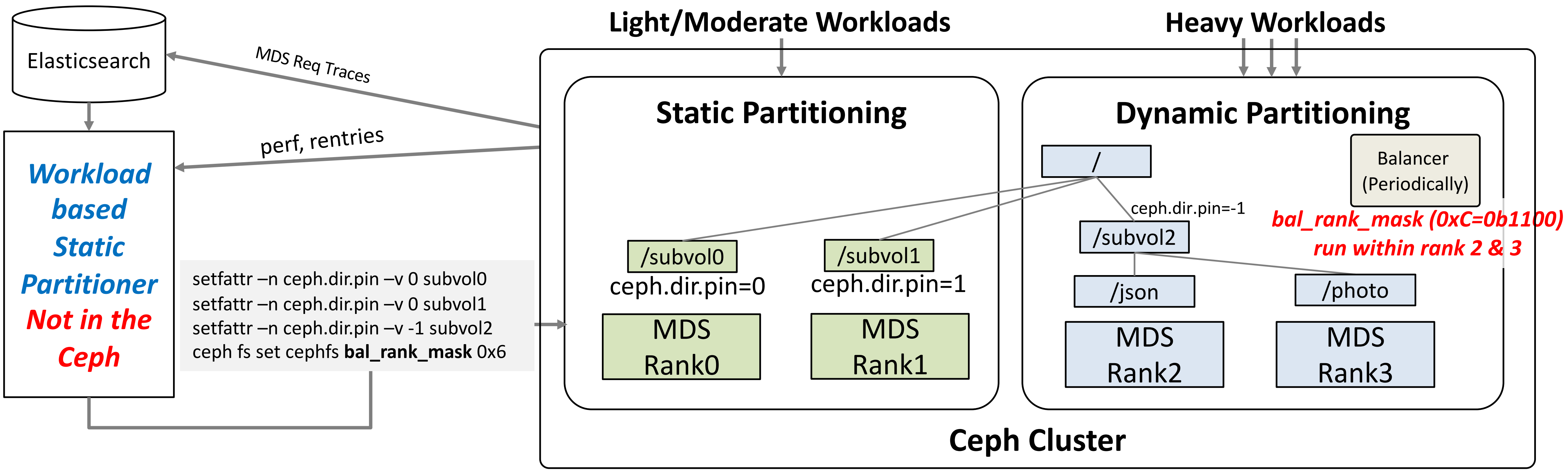
- Introduction and background
- CephFS subtree partitioning
- Combining static and dynamic partitioning schemes
  - Workload based static partitioner with bal_rank_mask
  - A new CephFS MDS balancer with ceph.dir.bal.mask
- Future works and conclusions

Our Idea: Combine Two Approaches

***Static partitioning** for moderate workloads, while **dynamic partitioning** for heavy workloads*



Workload Based Static Partitioner with bal_rank_mask



```
setfattr –n ceph.dir.pin –v 0 subvol0
setfattr –n ceph.dir.pin –v 0 subvol1
setfattr –n ceph.dir.pin –v -1 subvol2
ceph fs set cephfs bal_rank_mask 0x6
```

- ***Workload based static partitioner*** pins subvolumes
  - Workload calculation based on working set, rentries, and performance
  - Rarely or manually conducted if loads are uneven or latencies get higher
- ***bal_rank_mask*** enables the balancer to dynamically rebalance unpinned subtrees within particular active MDS ranks PR 43284

How to use bal_rank_mask

Change from 0b**111000** to 0b**000111**

```
$ ceph fs set cephfs bal_rank_mask 0x38
setting the metadata balancer rank mask to 0x38
$ sleep 180
$ ceph fs set cephfs bal_rank_mask 0x7
setting the metadata balancer rank mask to 0x7
```

```
$ ceph fs status
cephfs - 8 clients
======
RANK  STATE   MDS     ACTIVITY     DNS    INOS   DIRS   CAPS
 0    active  mds001  Reqs:    0 /s   5259   5095    84     14
 1    active  mds017  Reqs:    0 /s   463k   463k  31.1k    21
 2    active  mds018  Reqs:    0 /s   295k   295k  18.3k  38.3k
 3    active  mds016  Reqs:    2 /s  53.9k  53.9k  3573   53.6k
 4    active  mds015  Reqs:  405 /s  65.4k  65.5k  4018   65.0k
 5    active  mds010  Reqs:  398 /s  81.9k  81.9k  4906   81.8k
```

```
$ ceph fs status
cephfs - 8 clients
======
RANK  STATE   MDS      ACTIVITY     DNS    INOS   DIRS   CAPS
 0    active  mds001  Reqs:  364 /s  75.8k  75.4k  5943   70.9k
 1    active  mds017  Reqs:  259 /s   588k   587k  39.7k   123k
 2    active  mds018  Reqs:  366 /s   395k   394k  26.5k   137k
 3    active  mds016  Reqs:    5 /s    678    683    637     36
 4    active  mds015  Reqs:    8 /s    504    509    500     83
 5    active  mds010  Reqs:    4 /s    970    975    831     84
```

Evaluation Environment

- Ceph Pacific 16.2.10 (integrated with our PRs) installed in Rocky8
  - bal_rank_mask - PR 43284 (merged)
  - MDS QoS Scheduler - PR 38506 (ready to review)
- VDBench tool generates workloads
  - Each client has its own subvolume (e.g., /volumes/_nogroup/$subvol)

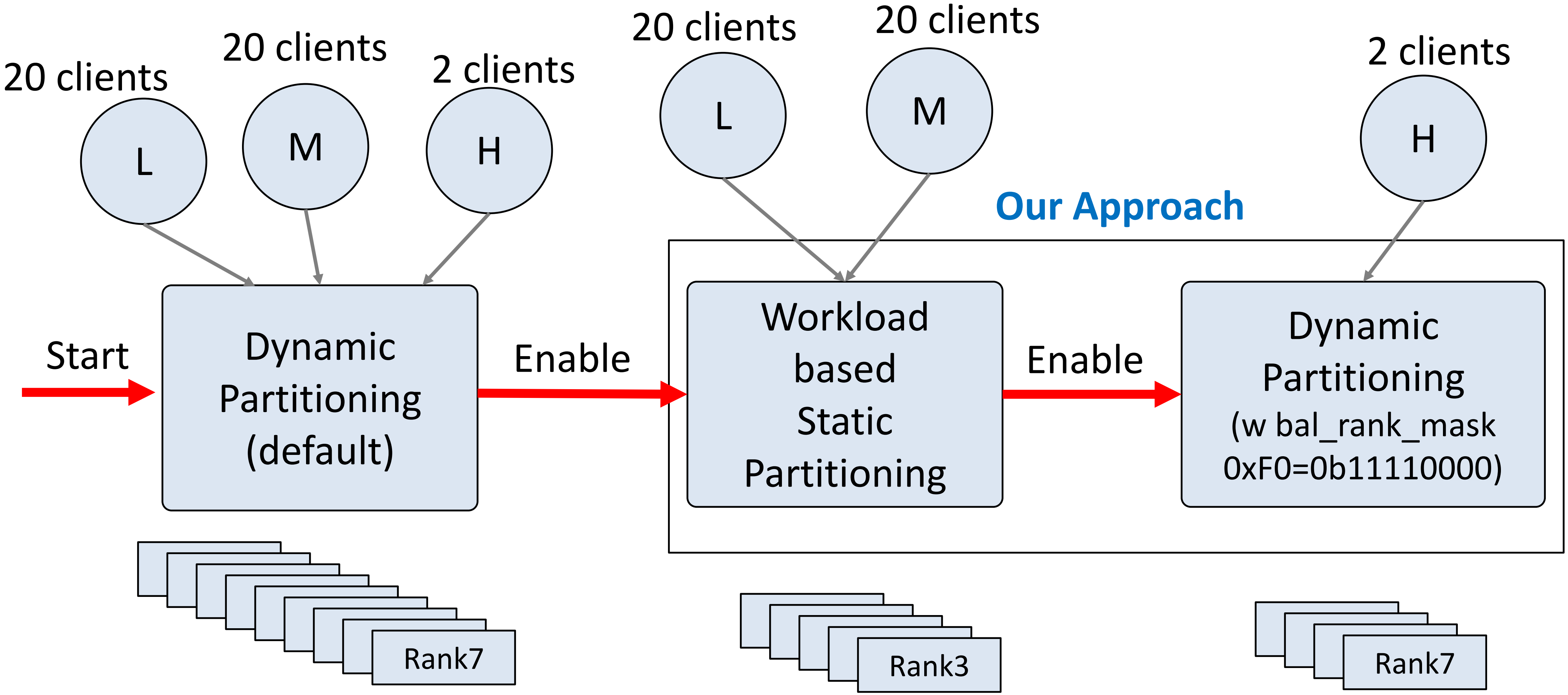
Workload Spec.

Type	Files	File Size	Clients	Threads per Client
Light	50K	4K	20	1
Moderate	500K	4K	20	1
Heavy	5,000K	4K	2	16

HW Spec.

Type	**Server Spec**	**Count**
MDS	4 * vCPU 32GB RAM	active 8 (mds_cache_memory_limt: 12GB)
OSD	40 * pCPU 128GB RAM	6 Servers * 6 SATA SSDs = 36
Mon	2 * vCPU 4GB RAM	3
Mgr	4 * vCPU 8GB RAM	3
Client	4 * vCPU 8GB RAM	42
Network	10Gbps	

Evaluation Sequences

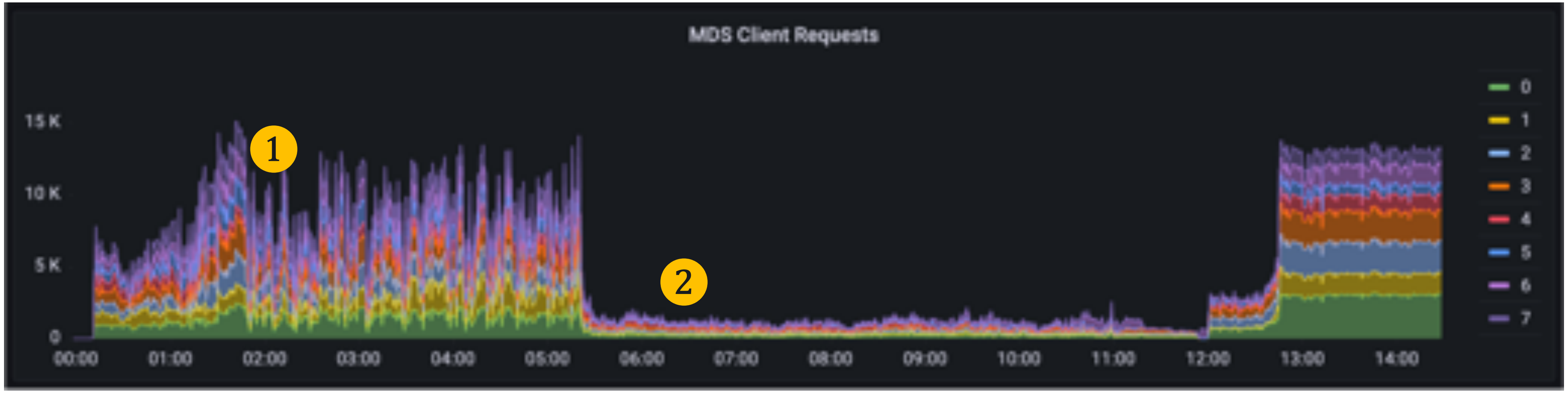

20 clients
L
20 clients
M
2 clients
H
Start
Dynamic Partitioning (default)
Enable
Rank7
20 clients
L
20 clients
M
Our Approach
Workload based Static Partitioning
Enable
Rank3
2 clients
H
Dynamic Partitioning (w bal_rank_mask 0xF0=0b11110000)
Rank7

Evaluation Results (1/3)

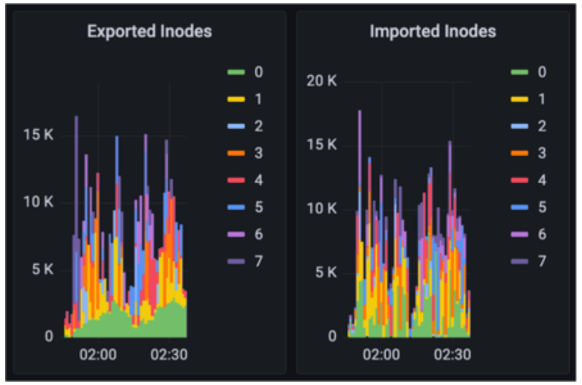
Balancer default

Our Static Partitioning

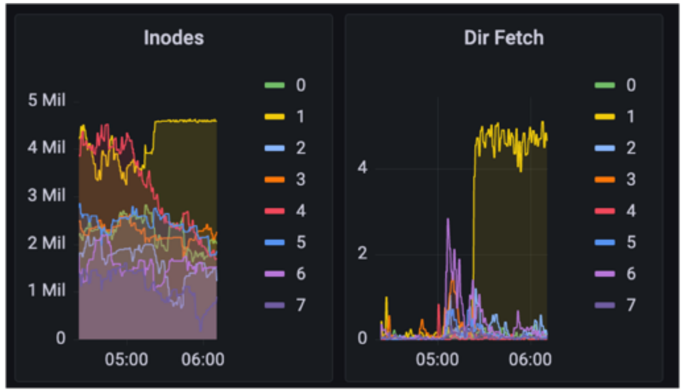
Dynamic Partitioning bal_rank_mask 0xf0



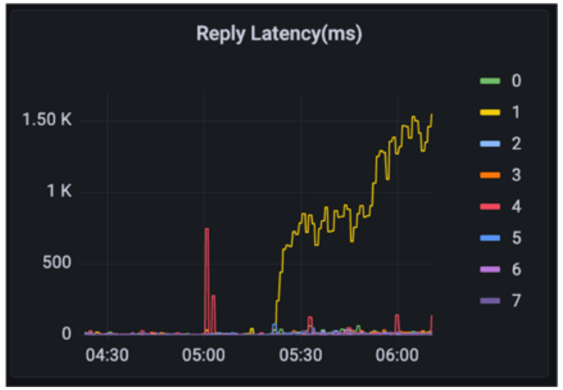
1 Performances of MDSs are fluctuated due to exporting/importing inodes



2 inodes are unevenly distributed rank1 keeps inodes more than 4mil.



Reply latencies highly increased

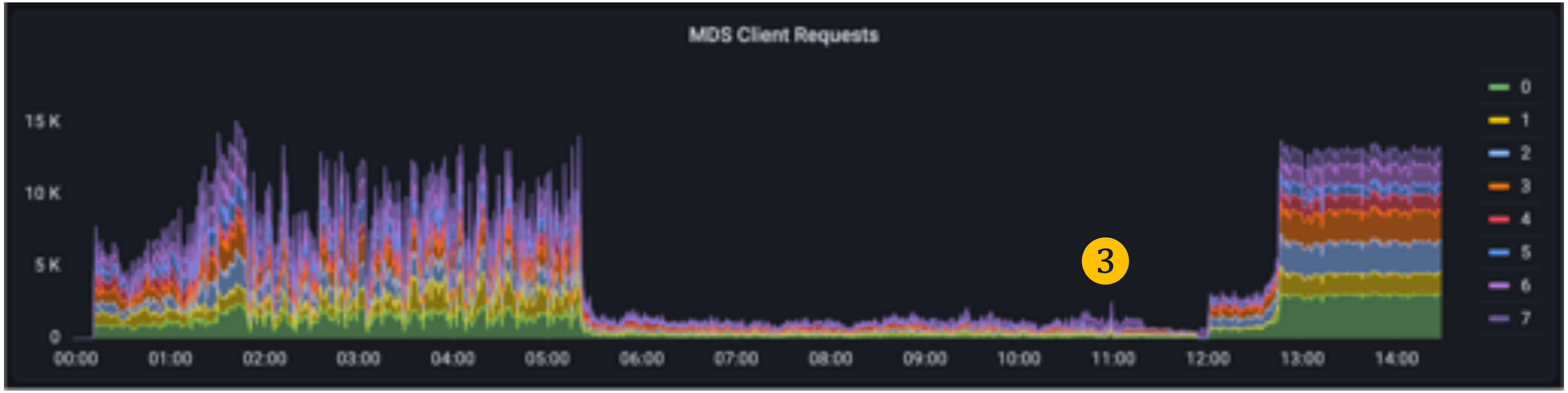


Evaluation Results (2/3)

Balancer default

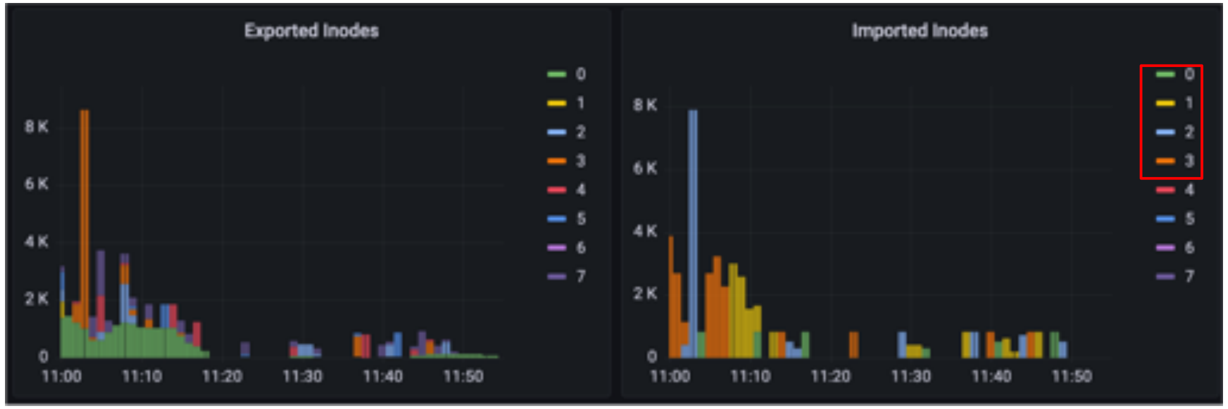
Our Static Partitioning

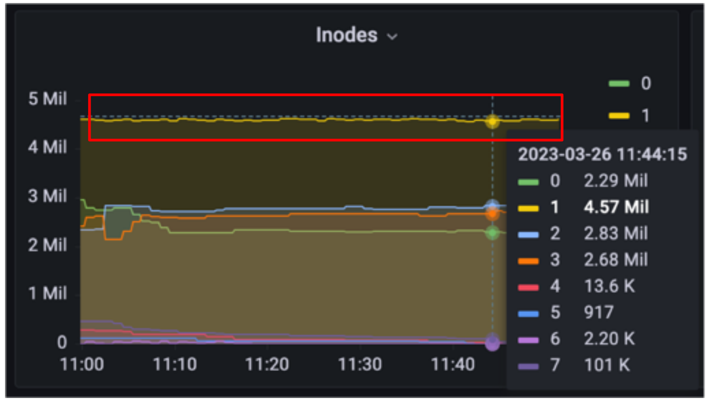
Dynamic Partitioning
bal_rank_mask 0xf0



3 subvolumes of light/moderate workloads are moved from all ranks to rank0~3

Performance is still not recovered as rank 1 has a lot of inodes



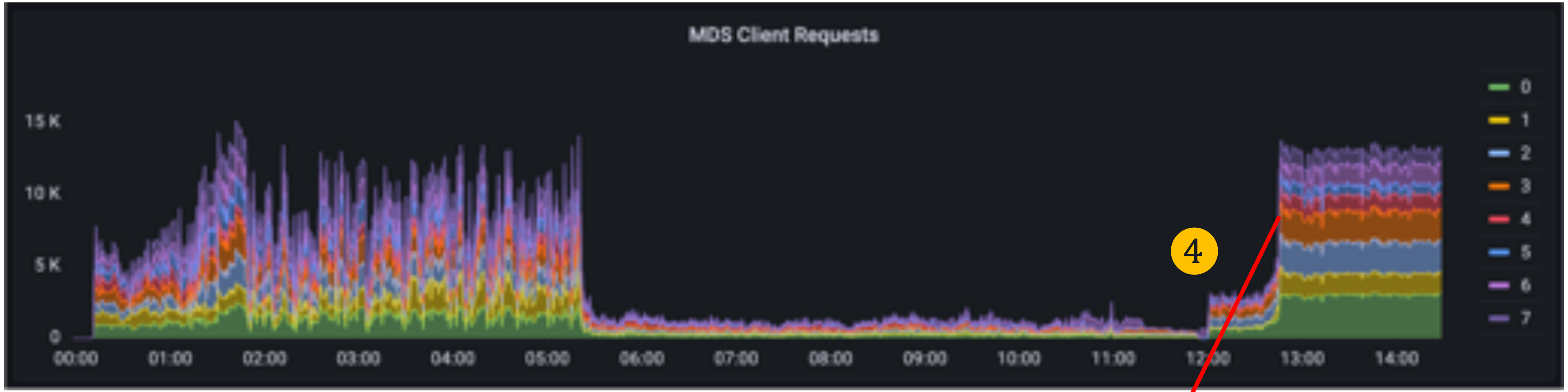


Evaluation Results (3/3)

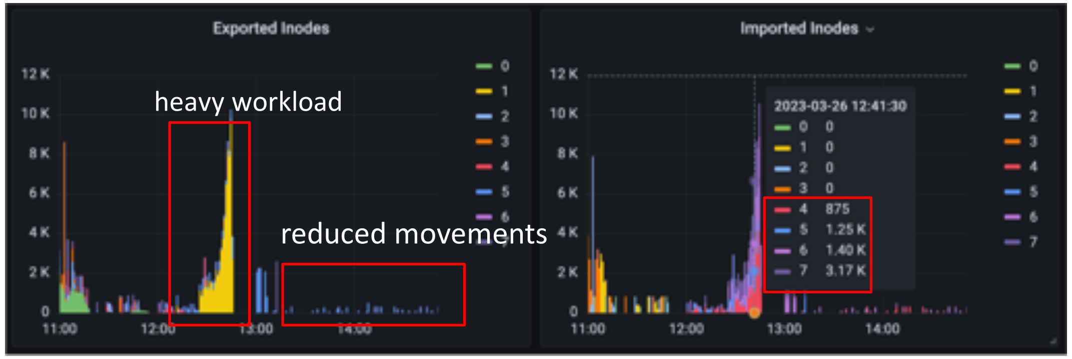
Balancer default

Our Static Partitioning

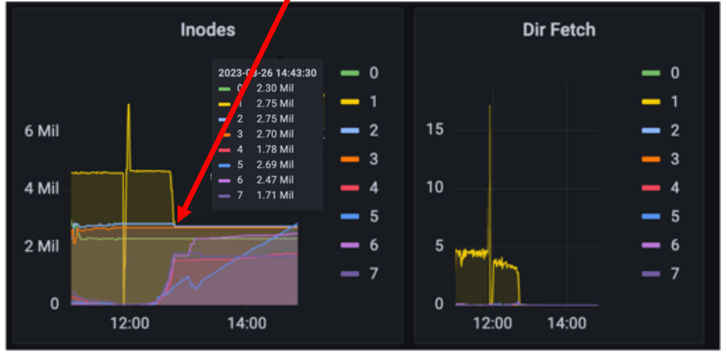
Dynamic Partitioning
***bal_rank_mask* 0xf0**



4 subvolumes of heavy workloads are migrated to rank 4~7



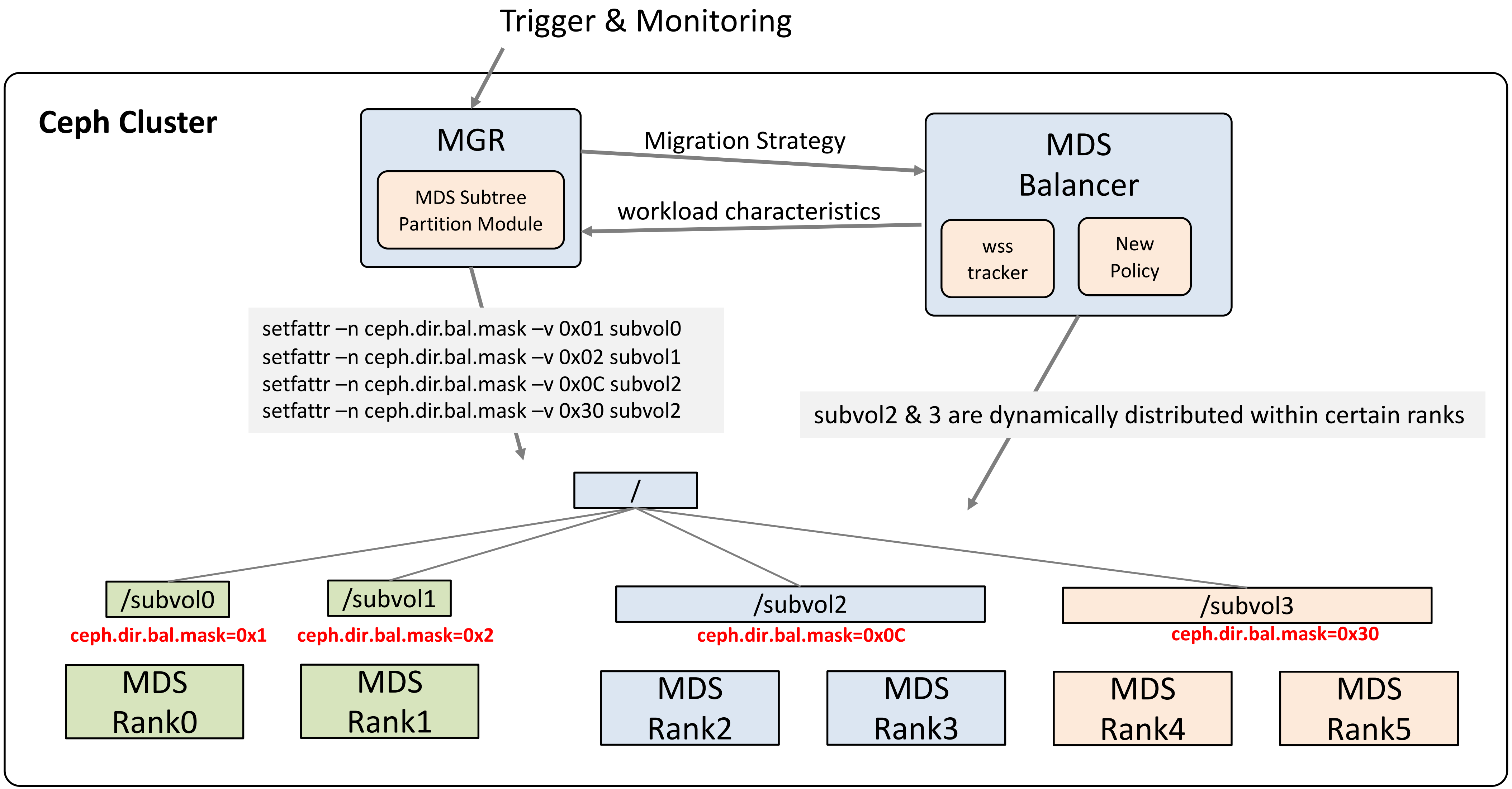
inodes are balanced and dir fetch count decreases and performance increases



Outline of Contents

- Introduction and background
- CephFS subtree partitioning
- Combining static and dynamic partitioning schemes
  - Workload based static partitioner with bal_rank_mask
  - A new CephFS MDS balancer with ceph.dir.bal.mask
- Future works and conclusions

A New MDS Balancer with ceph.dir.bal.mask



```
setfattr –n ceph.dir.bal.mask –v 0x01 subvol0
setfattr –n ceph.dir.bal.mask –v 0x02 subvol1
setfattr –n ceph.dir.bal.mask –v 0x0C subvol2
setfattr –n ceph.dir.bal.mask –v 0x30 subvol2
```

subvol2 & 3 are dynamically distributed within certain ranks

Implementation

- rank mask option per subdir as a virtual extended attribute
  - A target subdir is dynamically within certain MDS ranks (e.g., rank0 and 1)

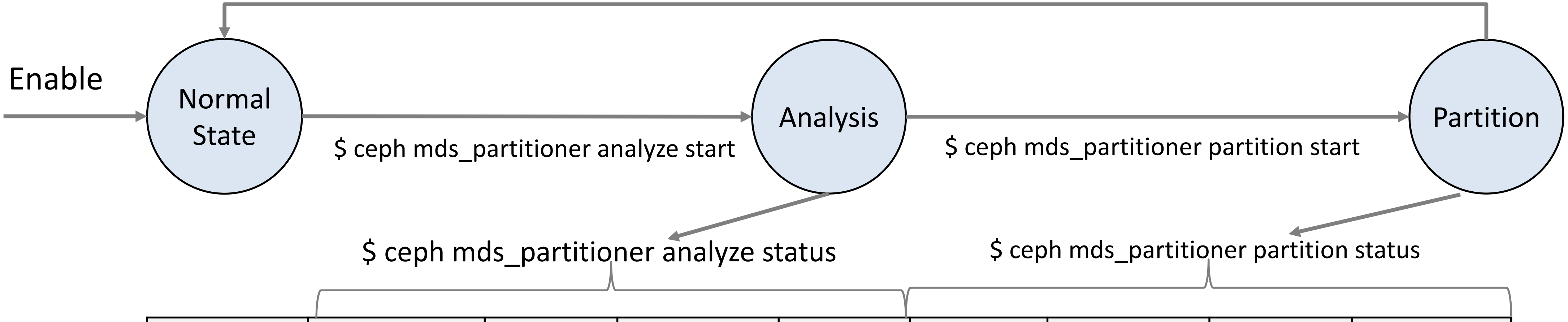
```
setfattr –n ceph.dir.bal.mask –v 0x3 /cephfs/home/yongseok
```

- MDS Subtree Partition Module in MGR

```
ceph mgr module enable mds_partitioner
ceph mds_partitioner analyze start   # analyze client workloads ontained from MDSs
ceph mds_partitioner analyze status # report analysis results and recommend optimal the number of MDSs
ceph mds_partitioner partition start # start partitioning
ceph mds_partitioner partition status # report partitioning status
```

- MDS Balancer modifications
  - Working set size tracker
  - Migrate subdirs based on ceph.dir.bal.mask values of subdirs
  - Minimize MDS slow requests

Example of Operation Flow



$ ceph mds_partitioner analyze start

$ ceph mds_partitioner partition start

$ ceph mds_partitioner analyze status

$ ceph mds_partitioner partition status

Name	wss	reqs	rentries	workload	Current Ranks	New Ranks	Migration Progress	Migration Status
Subvol1	1,000,000	1,203,030	50,000,000	339	0	0,1	20%	In-progress
Subvol2	700,000	500,000	1,000,000	137	1	2	0%	Ready
subvol3	100,000	5,000	5,000,000	21	2	2	100%	Done
subvol4	3,000	20,000	70,000	3	2	2	100%	Done
Total	1,803,000	1,728,030	56,070,000	500				

wss: working set size
reqs: requests
rentries: files + dirs

workload = (working_set_size / total_working_set *2
+ requests / total_requests * 2
+ rentries / total_rentries) * 100

Conclusions

- We employ CephFS as a shared file service in LINE’s cloud
- We compared static and dynamic partitioning schemes
  - Dynamic partitioning incurs metadata migration cost
  - Static partitioning present uneven workload distribution
- We presented how to combine both static and dynamic partitioning scheme effectively
- We will contribute our work on a new partitioning to the community

Thank you!

Any Questions?

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/
- ceph-linode for CephFS testing, Patrick Donnelly, https://github.com/batrick/ceph-linode
- CephFS with OpenStack Manila based on BlueStore and Erasure Code, 2018, https://cutt.ly/dc7Qnn7
- Revisiting CephFS MDS and mClock QoS Scheduler, Yongseok Oh, 2021, https://cutt.ly/A4s8AsC
- Ephemeral Pinning: A Dynamic Metadata Management Strategy for CephFS, Sidharth Anupkrishnan, 2020 https://www.youtube.com/watch?v=zimAEm_8efA
- Optimizing CephFS with Combining MDS QoS Scheduling and Static-Dynamic Subtree Partitioning, Yongseok Oh, https://ceph2023.sched.com/event/1JKas/optimizing-cephfs-with-combining-mds-qos-scheduling-and-static-dynamic-subtree-partitioning-yongseok-oh-jinmyeong-lee-line

VDBench Parameter Configs

client 01~02

```
fsd=fsd1,anchor=/mnt,depth=1,width=1,files=50000,size=4k
fwd=fwd1,fsd=fsd1,operation=getattr,xfersize=4k,fileio=sequential,fileselect=random,threads=1
fwd=fwd2,fsd=fsd1,operation=read,xfersize=4k,fileio=sequential,fileselect=random,threads=1
rd=rd1,fwd=(fwd1,fwd2),fwdrate=max,elapsed=86400,format=yes,interval=10
```

client 03~20

```
fsd=fsd1,anchor=/mnt,depth=1,width=1,files=50000,size=4k
fwd=fwd1,fsd=fsd1,operation=getattr,xfersize=4k,fileio=sequential,fileselect=random,threads=1
fwd=fwd2,fsd=fsd1,operation=create,xfersize=4k,fileio=sequential,fileselect=random,threads=1
fwd=fwd3,fsd=fsd1,operation=delete,xfersize=4k,fileio=sequential,fileselect=random,threads=1
fwd=fwd4,fsd=fsd1,operation=write,xfersize=4k,fileio=sequential,fileselect=random,threads=1
rd=rd1,fwd=(fwd1,fwd2,fwd3,fwd4),fwdrate=max,elapsed=86400,format=yes,interval=1
```

client 21~22

```
fsd=fsd1,anchor=/mnt,depth=1,width=10,files=50000,size=4k
fwd=fwd1,fsd=fsd1,operation=getattr,xfersize=4k,fileio=sequential,fileselect=random,threads=1
fwd=fwd2,fsd=fsd1,operation=read,xfersize=4k,fileio=sequential,fileselect=random,threads=1
rd=rd1,fwd=(fwd1,fwd2),fwdrate=max,elapsed=86400,format=yes,interval=10
```

client 23~40

```
fsd=fsd1,anchor=/mnt,depth=1,width=10,files=50000,size=4k
fwd=fwd1,fsd=fsd1,operation=getattr,xfersize=4k,fileio=sequential,fileselect=random,threads=1
fwd=fwd2,fsd=fsd1,operation=create,xfersize=4k,fileio=sequential,fileselect=random,threads=1
fwd=fwd3,fsd=fsd1,operation=delete,xfersize=4k,fileio=sequential,fileselect=random,threads=1
fwd=fwd4,fsd=fsd1,operation=write,xfersize=4k,fileio=sequential,fileselect=random,threads=1
rd=rd1,fwd=(fwd1,fwd2,fwd3,fwd4),fwdrate=max,elapsed=86400,format=yes,interval=1
```

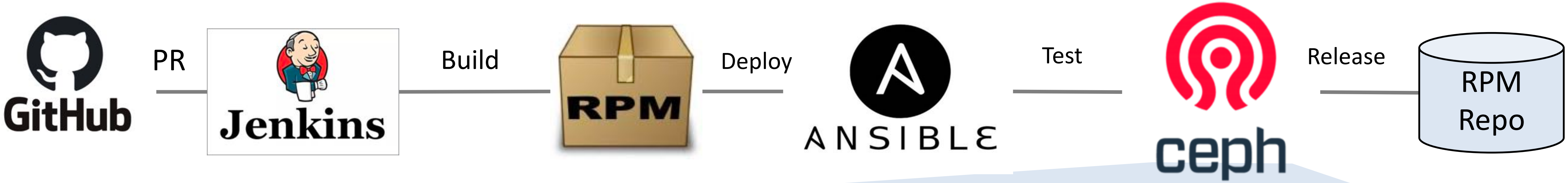
client 41

```
fsd=fsd1,anchor=/mnt,depth=1,width=100,files=50000,size=4k
fwd=fwd1,fsd=fsd1,operation=getattr,xfersize=4k,fileio=sequential,fileselect=random,threads=16
fwd=fwd2,fsd=fsd1,operation=read,xfersize=4k,fileio=sequential,fileselect=random,threads=16
rd=rd1,fwd=(fwd1,fwd2),fwdrate=max,elapsed=86400,format=yes,interval=10
```

client 42

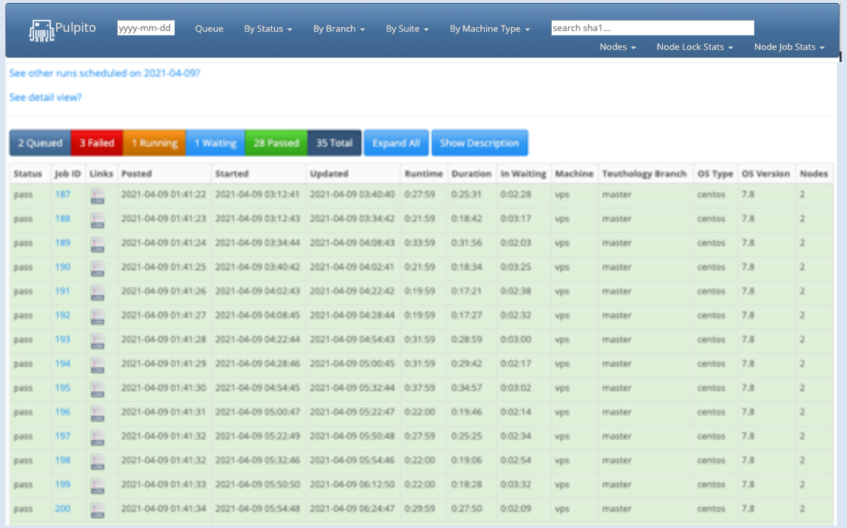
```
fsd=fsd1,anchor=/mnt,depth=1,width=100,files=50000,size=4k
fwd=fwd1,fsd=fsd1,operation=getattr,xfersize=4k,fileio=sequential,fileselect=random,threads=16
fwd=fwd2,fsd=fsd1,operation=create,xfersize=4k,fileio=sequential,fileselect=random,threads=16
fwd=fwd3,fsd=fsd1,operation=delete,xfersize=4k,fileio=sequential,fileselect=random,threads=16
fwd=fwd4,fsd=fsd1,operation=write,xfersize=4k,fileio=sequential,fileselect=random,threads=16
rd=rd1,fwd=(fwd1,fwd2,fwd3,fwd4),fwdrate=max,elapsed=86400,format=yes,interval=10
```

Ceph CI/CD System in LINE



Ceph Test Frameworks

[teuthology]



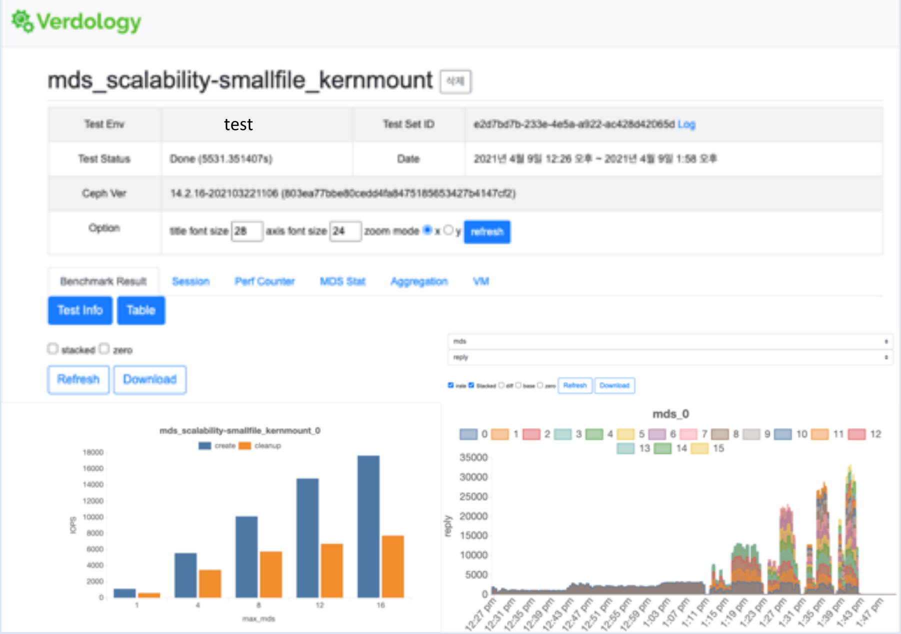
See other runs scheduled on 2021-04-09?

See detail view?

2 Queued | 3 Failed | 1 Running | 1 Waiting | 28 Passed | 35 Total | Expand All | Show Description

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

[verdology]



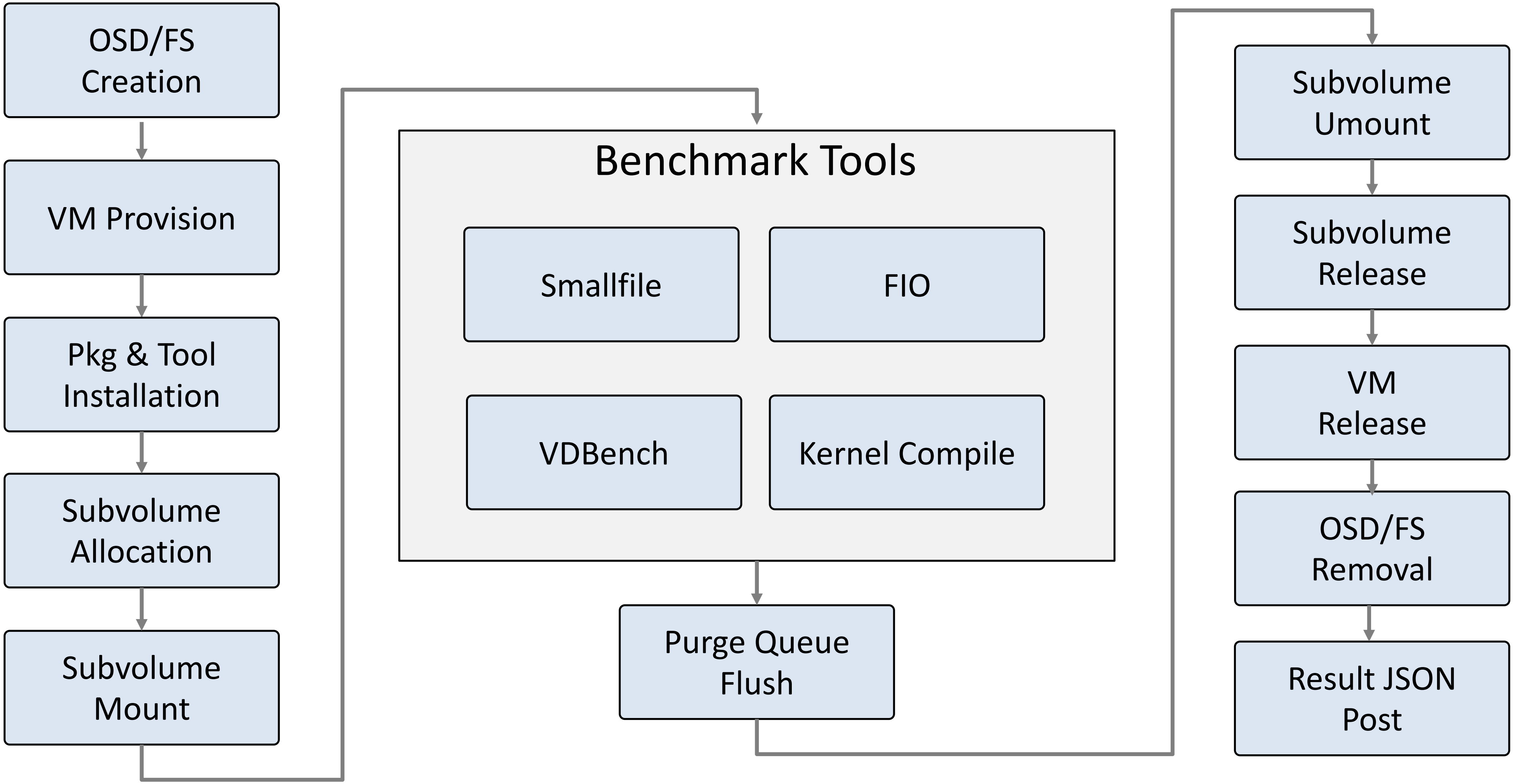
mds_scalability-smallfile_kernmount

Test Env	test	Test Set ID	e2d7bd7b-233e-4e5a-a922-ac428d42065d Log
Test Status	Done (5531.351407s)	Date	2021년 4월 9일 12:26 오후 ~ 2021년 4월 9일 1:58 오후
Ceph Ver	14.2.16-202103221106 (803ea77bbe80cedd4fa84751856534427b4147cf2)		
Option	title font size 28 axis font size 24 zoom mode ● x ○ y refresh		

□ stacked □ zero

☑ iops ☑ Stacked □ diff □ base □ zero

CephFS Test Flow of Verdology


OSD/FS Creation
VM Provision
Pkg & Tool Installation
Subvolume Allocation
Subvolume Mount
Benchmark Tools
Smallfile
FIO
VDBench
Kernel Compile
Purge Queue Flush
Subvolume Umount
Subvolume Release
VM Release
OSD/FS Removal
Result JSON Post