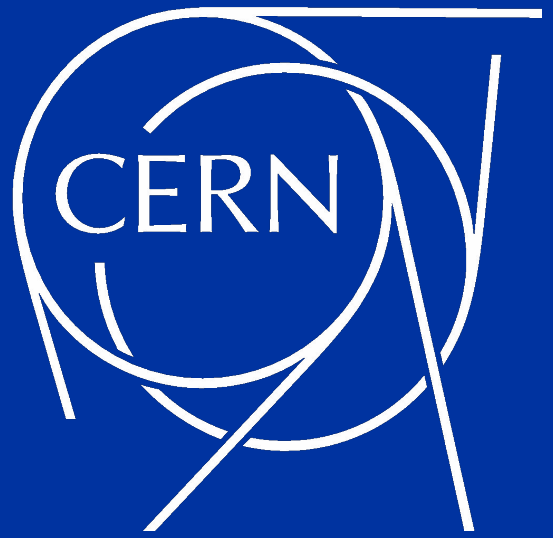
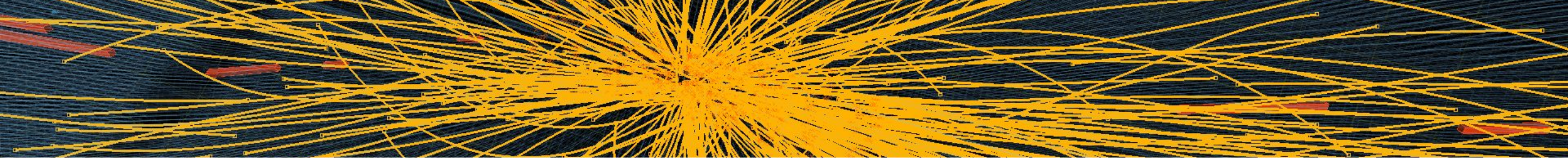

CERN

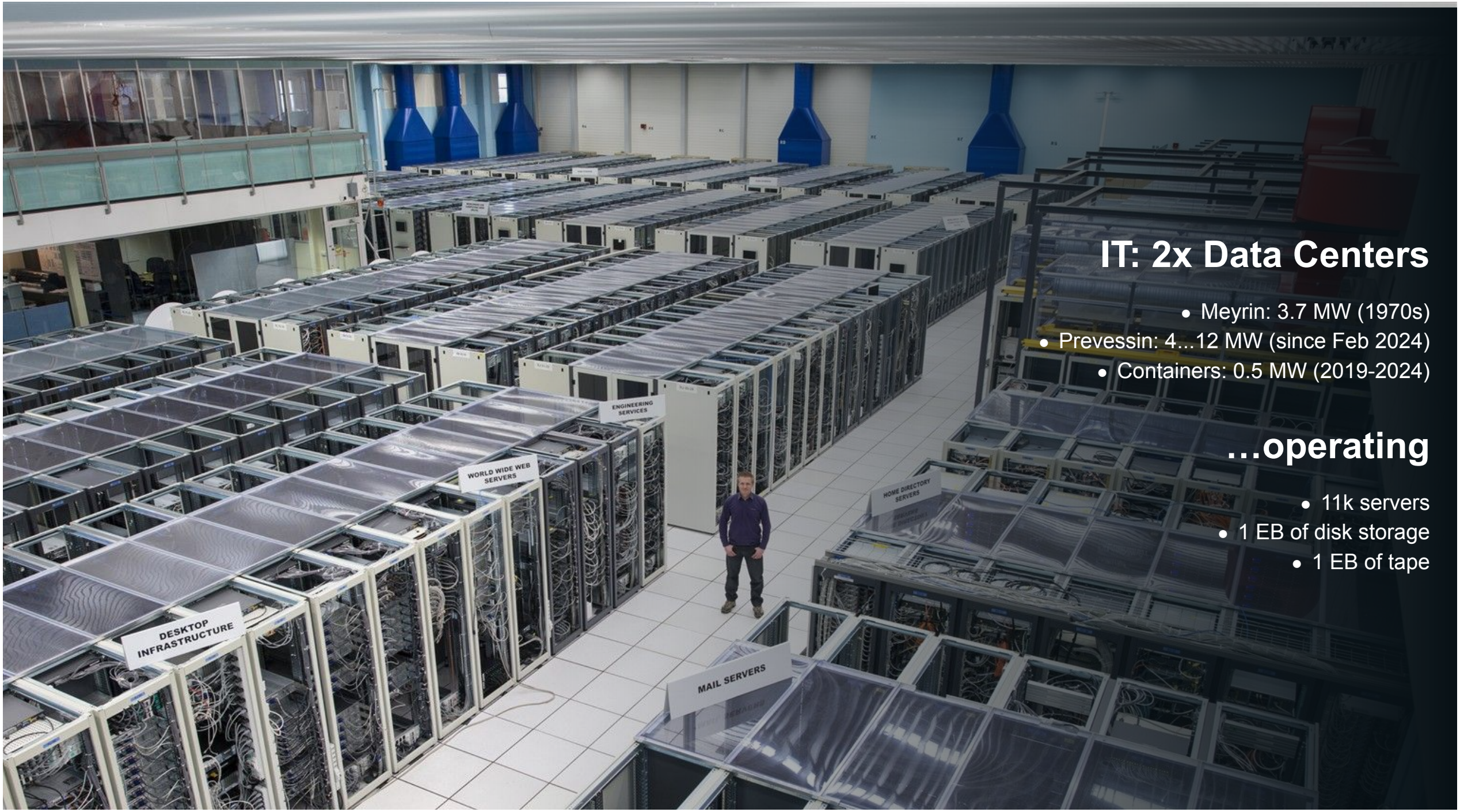


Modernizing Ceph Deployments at CERN

CephFS and Object Storage Across Data Centres



Enrico Bocchi
STORAGE & DATA GROUP
CEPH TECHNICAL LEAD



IT: 2x Data Centers

- Meyrin: 3.7 MW (1970s)
- Prevessin: 4...12 MW (since Feb 2024)
- Containers: 0.5 MW (2019-2024)

…operating

- 11k servers
- 1 EB of disk storage
- 1 EB of tape

A Brief Ceph History at CERN

2013:	300TB proof of concept, 3 PB in production for RBD (QEMU) in OpenStack
2014:	Implementation of Intel ISA Erasure Coding plugin
2016-17:	3PB to 6PB with no downtime
2018:	S3 + CephFS in production
2019:	Optimizing CephFS for HPC applications
2020:	Backup cluster in 2nd location (S3)
2021:	New media types, RBD Storage Availability Zones
2023:	kernelRBD in production, RBD Backups
2024:	2nd Datacenter online for Disaster Recovery, Cephalocon at CERN
2025:	100+ PB, 10k OSDs, 20+ production clusters in 3 locations





Outline

1. CephFS

- Availability Zones & Dedicated File Systems
- Integration with K8s Orchestrators
- Backups

2. CephFS: multiFS

- Operations & Disaster Recovery

3. CephFS: Metadata Balancing

- Strategies and Benchmarking

4. Cross-DC Storage

- CephFS metadata and clients locality
- Objects stores multisite and backup lifecycle

1

CephFS

Availability Zones, Dedicated FS, and Backups

CephFS at CERN

What for?

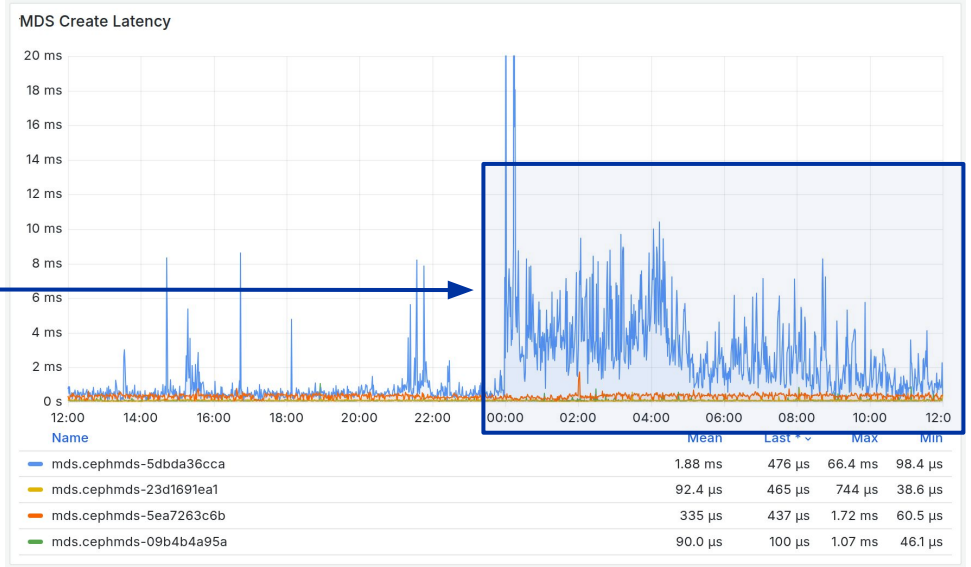
- Persistent Storage for K8s: Enterprise groupware, project mgmt tools, code repositories, web-hosting, …
- HPC Staging / Scratch spaces: Hyperconverged storage on HPC cluster, `LazyIO` as option
- LinuxSoft / Linux at CERN: Distribution of RPMs for all Linux nodes at CERN + `Koji` software building
- Windows DFS Replacement: CephFS exported over SMB for project spaces
- Prototyping $HOME filestore with CephFS via NFS export

Clusters as of today:

- 4x General Purpose:
  - HDD (with metadata on SSD) and full-flash (meta and data)
  - 1 FS, multi-active MDS (Balancer disabled, manual static pinning of subvolumes), no snapshots
  - Integrated with OpenStack Manila: Users ask for quota and get a slice of the FS
- 4x Targeted use cases:
  - 1x HPC staging/scratch space
  - 1x SMB export of CephFS for Windows DFS replacement
  - 2x early adoption of dedicated FS – more later…

Concerns from daily CephFS operations

1. **Performance / QoS Shaping**
   - No throttling possible for metadata requests by clients
   - *“Something has happened…”*



2. **Poisonous patterns from clients**
   - Stuck requests due to lock contention among clients
     - Might lead to MDS not trimming journal, which is bad…
     - Evicting the client(s) at fault solves the problem, but…
   - MDS might crash when evicting
     - In `"finisher"`, if client has outstanding `"batch getattr"` requests (70624, 70769)
     - Further down the road (`num_rdlock`, `num_pins`), attempting to fix the crash in `"finisher"` (72941, 73006)
   - Other ranks crashing while failed rank is in `rejoin` (62036)

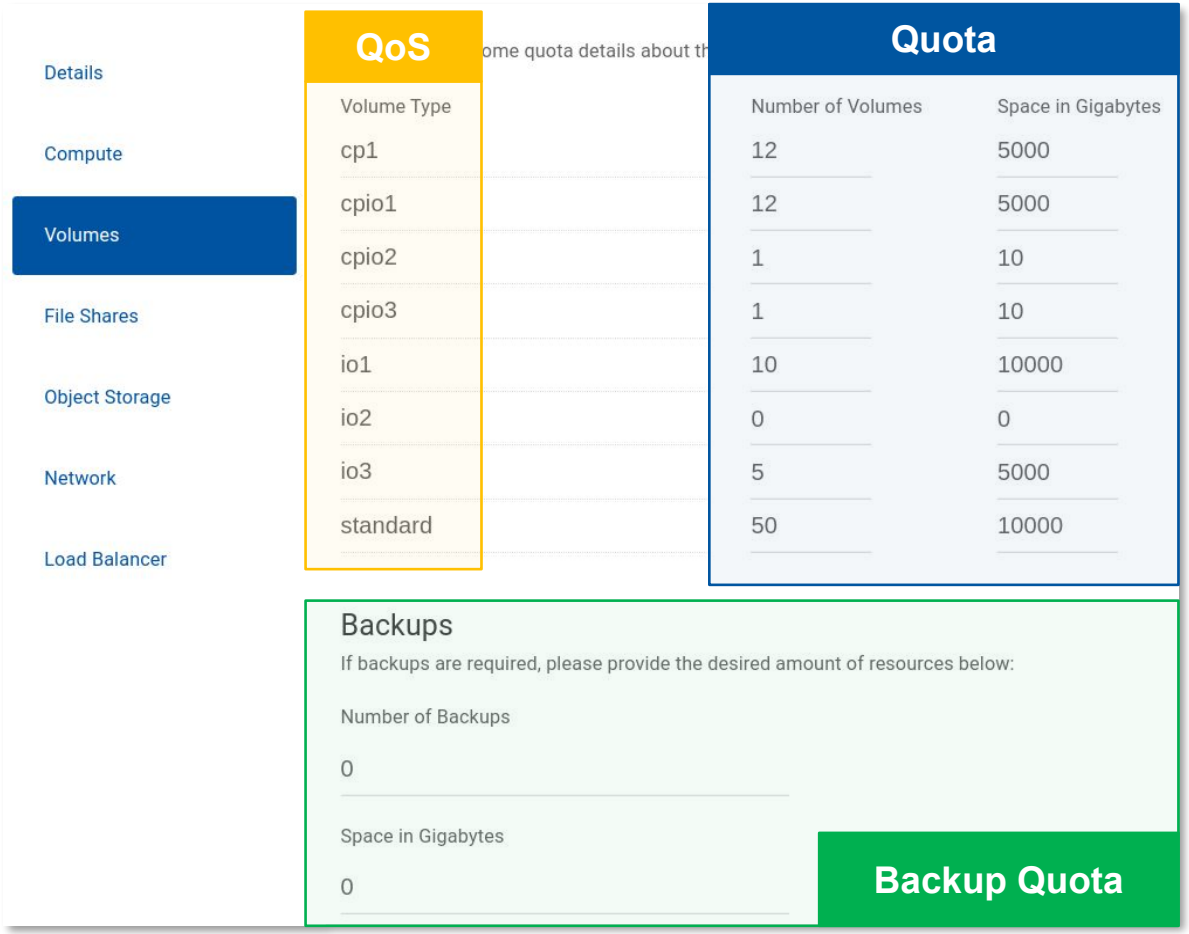
3. **Backups**
   - Walking large directory hierarchies is expensive for the MDS: Cache thrashing, cap locking rtts
   - Even with snapshots, MDS is exposed to extra load for stat/read operations

Small Digression on RBD Service at CERN

RBD service at CERN
has a rather complete feature set:

- Client throttling as per QoS profiles (IOps, BW) mapped to HDD/NVMe, Replica/EC pools
- Independent clusters providing the same QoS ⇒ RBD Availability Zones
- Cheap CoW Snapshots
- Efficient Backups with `fast-diff+object-map`

Goal: Revisit CephFS service delivery
taking inspiration from RBD maturity



Moving Away from Large CephFS

OpenStack Manila (shared file system service) supports:

- Multiple *"share types"*, e.g. "CephFS", "CephFS Vancouver", "CephFS Canada"
- Multiple storage backends for each share type

Dedicated File Systems	CephFS Availability Zones

• Reserved for a specific application or group
• New Ceph FS (dedicated meta+data pool and MDS)
  • Can be co-hosted on existing HW with other FSes
• Mapped to specific share type in Manila
  • E.g., "CephFS Vancouver"
  • Access granted to selected OpenStack projects
• In production for a few applications, more to come
• Major disadvantage: Data has to be copied over
	• Available to general audience
	• New Ceph cluster with FS on top
	  • Dedicated HW resources hosted in zone of the DC where there is no CephFS already
	• Bound to to general purpose share type ("CephFS"); Configured as Availability Zone
	• Demonstrated to work, not production (yet?)
	• Onboarding of use cases impractical

Integration with K8s Orchestrators

A large fraction of CephFS is used by applications deployed via container orchestrators

⇒ Ease of use is essential: `helm install <myapp>`

- Majority of users simply rely on default StorageClass, or StorageClass for faster storage
- Majority of volumes are dynamically provisioned

Requires coordination with two components:

1. **Provisioner** – `cephfs.manila.csi`
   - Interfaces with OpenStack Manila APIs for creation and mgmt of volumes
2. **Mounter** – `ceph-csi`
   - The actual CephFS client, fuse or kernel

Integration with K8s Orchestrators

Kudos to Devs for fixing:
- fsName parsing in Manila (2994)
- Fuse mount options (5643)

Target Dedicated File Systems via custom StorageClass definition:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: manila-cephfs-vancouver
allowVolumeExpansion: true
parameters:
  type: "CephFS Vancouver"
  cephfs-mounter: "fuse"
provisioner: cephfs.manila.csi.openstack.org
```

This is an OpenStack Manila Share type:
- Manila is configured for relevant storage backend
- Exposes `__mount_options='fs=vancouver'` among properties

Parsed and propagated to the mounter:
- fuse: client_fs=vancouver
- kernel: fs=vancouver

CephFS Availability Zones are less obvious to expose:
- Manila share type is the same for different AZs, though multiple storage backends exist
- Provisioner does not seem to be instrumented for defining AZs when creating, nor Manila CSI to expose this info
- Mounter should not even know, as long as mount params are correct
- Overlaying K8s orchestrator should leverage by implementing topology-aware volume provisioning (5177)

CephFS Backups: cback

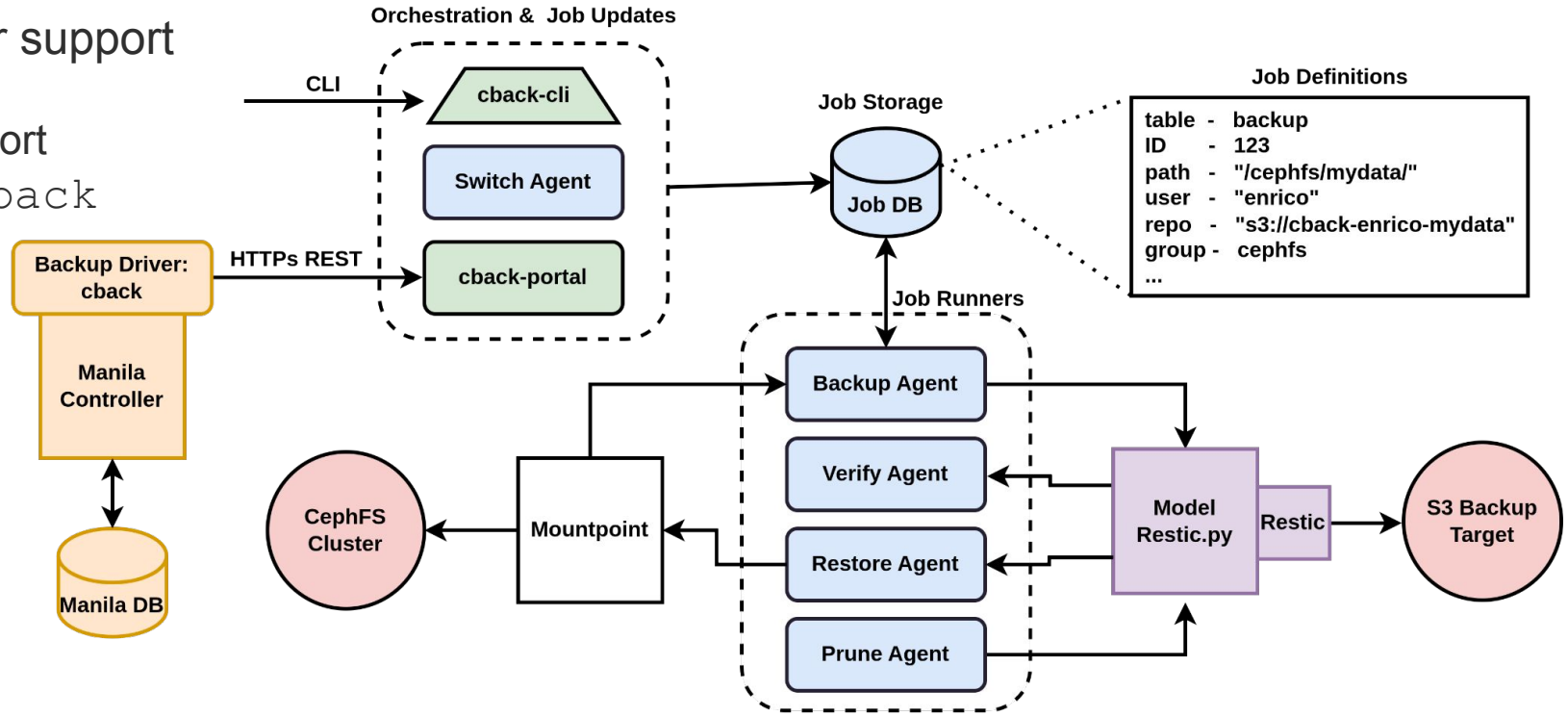
Credit to **Zac Goggin**

cback: Backup orchestrator based on Restic

- Creation and management of backups from any mounted FS to S3 bucket

Recently merged into OpenStack Manila:

- Improved 3rd party backup driver support
  - Added out-of-place restores
  - Added driver-driven progress report
- Developed a custom driver for `cback`
- Allows for user-driven backups and restores via Manila
- Controls backup/restore jobs progress and state handling



2

CephFS

Multiple File Systems (multiFS)

Multiple File Systems (multiFS)

Independent file systems hosted on one cluster

- Must be explicitly enabled with `` `ceph fs flag set enable_multiple true` ``
- One additional MDS required (+ stand-by); New metadata and data pools
- Possible to tune cephx keys caps so clients have restricted view on FSes and communication to MDS

Why multiFS instead of multi-active MDS?

- “Concerns from daily CephFS operations”: QoS/Performance, Poisonous patterns, Backups
- Recovery procedures require taking the entire FS down, not only the affected rank(s)
- Global data structures tracking hard links in snapshots (global `SnapRealm`)
- Metadata balancing strategy across ranks not obvious (more later…)

- Independent FSes are isolated entities, but can be co-hosted on same HW resources
  - Better clients and workloads isolation, reduced impact in case of issues
  - Little extra cost: Additional MDS(es) for the new FS

Multiple File Systems (multiFS)

Usual `ceph fs status`
shows all FSes, stats, and pools

```
cephfs - 107 clients
======
RANK  STATE                MDS                    ACTIVITY     DNS    INOS   DIRS   CAPS
 0    active  cephfs.cephp046-3599f2b213.repdoi  Reqs:    8 /s  2155k  2076k   217k   184k
    POOL       TYPE     USED  AVAIL
cephfs_meta  metadata  36.2G  61.1T
cephfs_data    data      15T  61.1T
other - 12 clients
=====
RANK  STATE                MDS                    ACTIVITY     DNS    INOS   DIRS   CAPS
 0    active  other.cephp046-3ee53cf570.lniypm  Reqs:   93 /s   11.2k  11.0k   724   1755
   POOL       TYPE     USED  AVAIL
other.meta  metadata   749M  61.1T
other.data    data    2846M  61.1T
default - 31 clients
=======
RANK  STATE                MDS                    ACTIVITY     DNS    INOS   DIRS   CAPS
 0    active  default.cephp046-a45ed014b1.arinen  Reqs: 1127 /s   349k   348k  23.3k  36.5k
    POOL       TYPE     USED  AVAIL
default.meta  metadata  1204M  61.1T
default.data    data     201G  61.1T
          STANDBY MDS
cephfs.cephp046-1f4acae3c1.lfbpbf
MDS version: ceph version 18.2.7 (6b0e988052ec84cf2d4a54ff9bbbc5e720b621ad) reef (stable)
```

Operating multiFS 101

1. Tuning options with FS granularity:
   - FS flags via `ceph fs set` on each FS
   - MDS config via `mds.<fname>` mask

```
$ ceph config set mds.cephfs set …
     mds_cache_memory_limit
     mds_max_caps_per_client
```

2. Upgrades:
   - `cephadm` stops all MDS and deploys new version
   - We have upgraded MDS one by one (e.g., one FS down at a time) and worked – Dangerous? MDS capabilities?

MultiFS Disaster Recovery

Two main Disaster Recovery procedures known and documented

1. Recover FS after Monitor store loss ⇒ docs.ceph.com/.../recover-fs-after-mon-store-loss

```
The steps to recover a multiple active MDS file system or multiple file systems are yet to be
identified. Currently, only the steps to recover a single active MDS file system with no
additional file systems in the cluster have been identified and tested.
```

🎉 **It works!**

2. Metadata repair tools ⇒ docs.ceph.com/.../disaster-recovery-experts/

🎉 **These also work!**

MultiFS Recovery: Caveat on Default FS

FSMap stores which is the "*Default FS*" among the available ones

- Reported as `"legacy client fscid"` by `` `ceph fs dump` ``, or as `"default_fscid"` when using json output
- This is the ID of the FS, not the FS name

```
# ceph fs dump
e72
legacy client fscid: 1
{...}

Filesystem 'cephfs' (1)
fs_name    cephfs
epoch      11
flags      12 joinable allow_snaps allow_multimds_snaps
{...}
```

Clients must specify `` `fs=<fsname>` `` when mounting a non-default FS

```
mount -t ceph rectest.cern.ch:6789:/ /cephfs -o name=client01,secret=123XYZ,fs=vancouver,...
```

- If they don't, they will attempt to mount the default FS
- It is common practice to call the default FS *"cephfs"*
  - This is likely the first one created on a cluster, or the only existing one

MultiFS Recovery: Caveat on Default FS

Problem: The first recovered file system upon MON store loss becomes the default one

- We have thousands of clients not specifying the FS name…
  - Upon recovery, they will all try to mount the default FS assuming to find *"cephfs"*
  - If the default is not *"cephfs"*, they will bail out with "mount error 2 = No such file or directory"

Solution:

- Restore *"cephfs"* first :)
- Set the default filesystem with `` `ceph fs set-default <fsname>` ``
  - It can be done online!
  - This will update the `"default_fscid"` value; Clients will successfully mount on the next attempt
- Make FS name always explicit when client mounts
  - The new mount syntax helps

```
mount -t ceph client01@cephfs=/ /cephfs -o mon_addr=rectest.cern.ch:6789,secret=123XYZ,...
```

- Use `` `ceph fs swap` `` to swap FS names and `fscid`s
  - Quite cumbersome: Requires FSes offline and two rounds of `` `fs swap` `` – `` `fs set-default` `` is much better

MultiFS Recovery: Metadata Tools

1. Journal manipulation

- `cephfs-journal-tool` supports multiFS and multiple ranks with `--rank=<fsname>:<rank>` option

```
cephfs-journal-tool --rank=myfs:{0..N} journal export myfs_N.journal.bin
cephfs-journal-tool --rank=myfs:{0..N} event recover_dentries summary
cephfs-journal-tool --rank=myfs:{0..N} journal reset
```

2. MDS tables manipulation

- `cephfs-table-tool` requires specifying FSname and rank as parameter

```
cephfs-table-tool myfs:{0..N} reset <session|snap|inode>
```

3. Missing metadata objects and alternate recovery pool

- `cephfs-data-scan` has optional FSname and pool parameters

```
ceph osd pool create myfs_recovery_meta
ceph fs new myfs_recovery myfs_recovery_meta myfs_data --recover --allow-dangerous-metadata-overlay
cephfs-data-scan init --force-init --filesystem myfs_recovery --alternate-pool myfs_recovery_meta
cephfs-data-scan scan_extents --alternate-pool myfs_recovery_meta --filesystem myfs_recovery --worker_n 0 --worker_m 4
```

3

CephFS

Metadata Balancing

Metadata Balancing Techniques

0. None

- Cranking up `max_mds` (`ceph fs set cephfs max_mds 2`) does not buy you anything
- All requests will still go to rank 0, with other ranks idle

1. Automatic Balancer

- Aka “the balancer”, default balancer, dynamic balancer – Not the MGR balancer module
- Relevant parameters: `mds_bal_mode`, `mds_bal_interval`, `bal_rank_mask`, split/merge/frag thresholds, …
- Known to be problematic for hotspots / flash crowds on large directories
- Disabled by default since Reef
  - To enable: `ceph fs set cephfs balance_automate true `

2. Manual Pinning

- Configure export pins to map a directory tree to a desired rank
- Set extended attribute: `setfattr -n ceph.dir.pin -v 1 /cephfs/dir/to/pin `
- Load balancing is left to the cluster administrator
- Since Luminous

Metadata Balancing Techniques

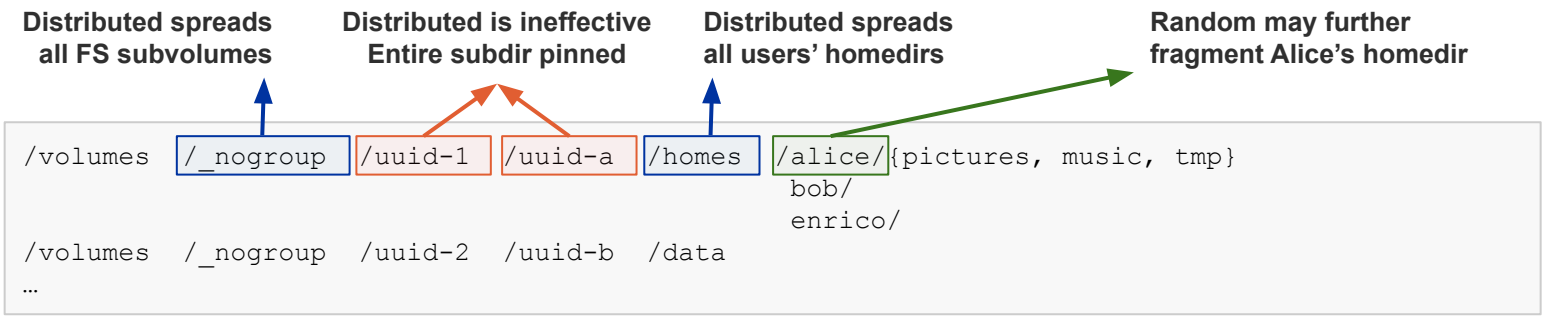
3. Ephemeral Distributed Pinning

- Distributes children across MDS ranks
- Set extended attribute: `setfattr -n ceph.dir.pin.distributed -v 1 /cephfs/homes`
  - Results in subdirectories (e.g., `/cephfs/homes/{alice, bob, enrico}`) being distributed as per hash of inode
- Introduced with Octopus, stable in Pacific

4. Ephemeral Random Pinning

- Set extended attribute: `setfattr -n ceph.dir.pin.random -v 0.01 /cephfs/homes`
  - Results in subdirectories being pinned 1% of the time

Some care required to decide where to set the extended attributes:



Metadata Balancing Measurements

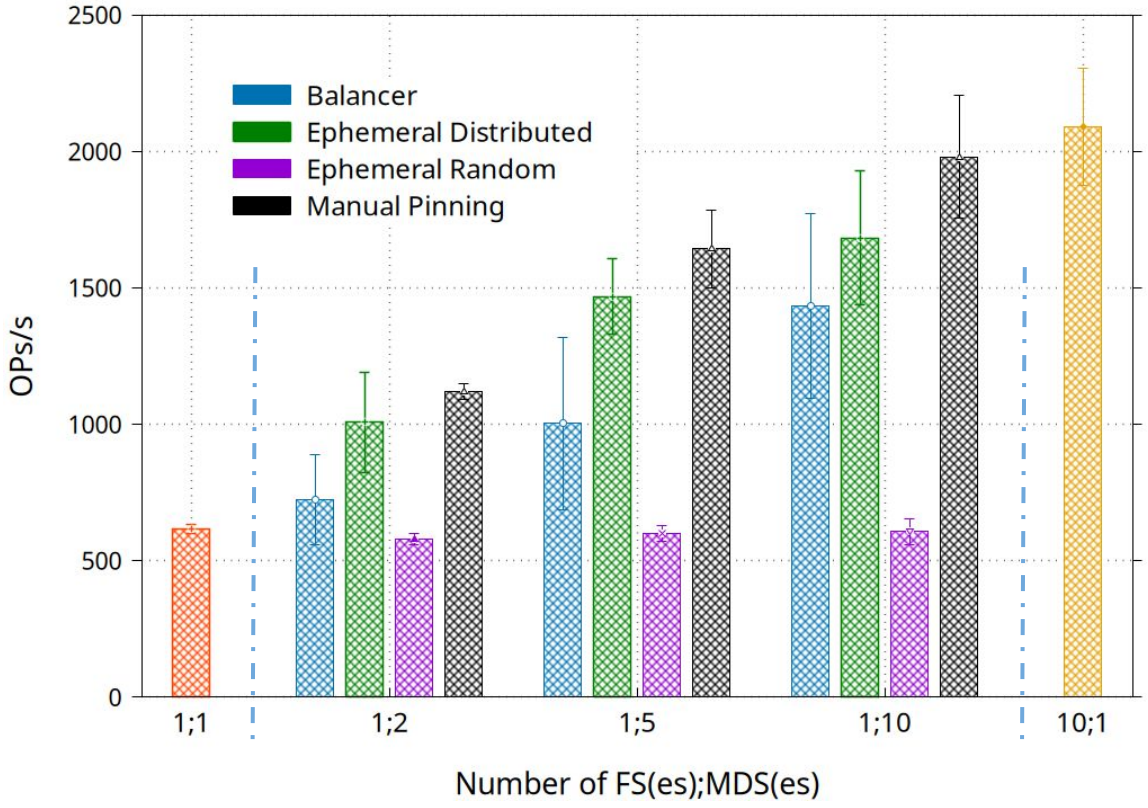
- **Test Instrumentation**
  - IO500 benchmarking suite; Focus on `mdtest`
    - Create/stat/read/rename/remove files and directories (cherry-pick each test *phase*)
    - Depth/branching factor of hierarchical structure; Items (files/dirs) per directory in tree
  - `mdtest_args='-P -C -i=1 -b=3 -z=4 -I=1084 -w 4096 -e 4096'`
  - `--run-cmd-before-phase="echo 3 >/proc/sys/vm/drop_caches; sleep 5"`

- **Test Infrastructure**
  - 15x servers (16 cores, 128 GB, 16x 0.96 SATA SSDs, 10Gbps); 10 clients (8 cores, 16 GB, 10Gbps)
  - AlmaLinux 9.6, Ceph Reef 18.2.7, managed by `cephadm`
  - All configuration parameters are left to the defaults, with the exception of
    - `osd_memory_target=4GB`; `osd_mclock_max_capacity_iops_ssd=25000`
    - `mds_cache_memory_limit=16GB`

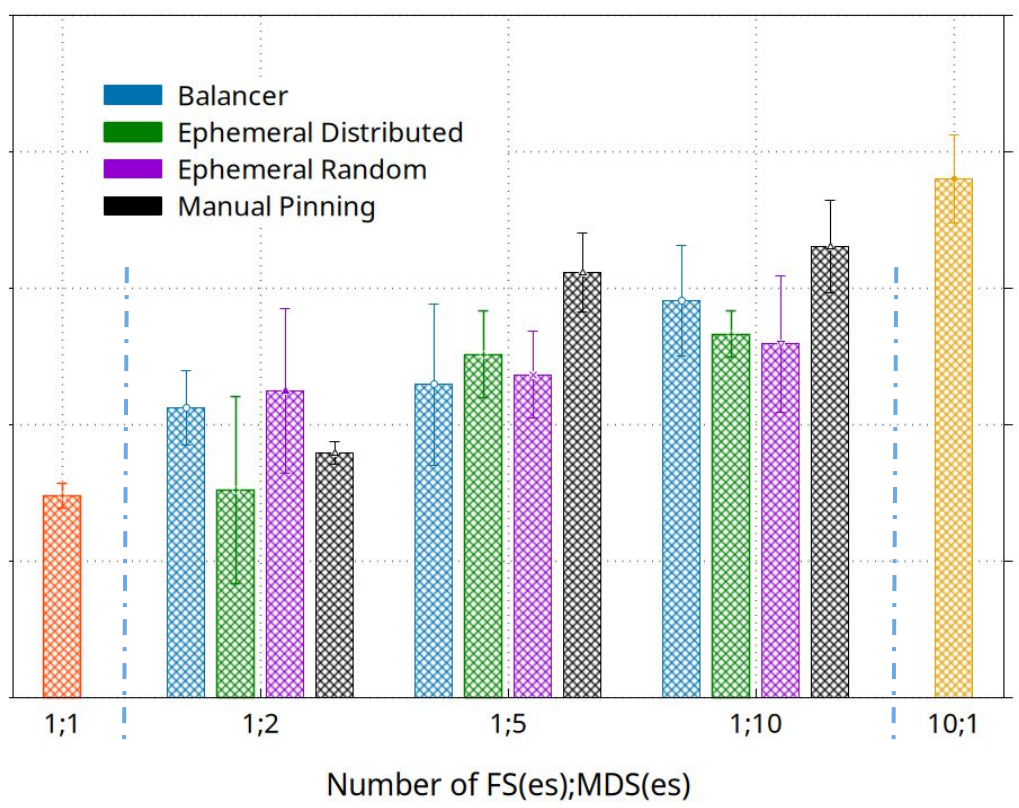
- **Purpose:**

  Characterize behaviour and performance of balancing techniques under known testing conditions.

Metadata Balancing Measurements: Directories

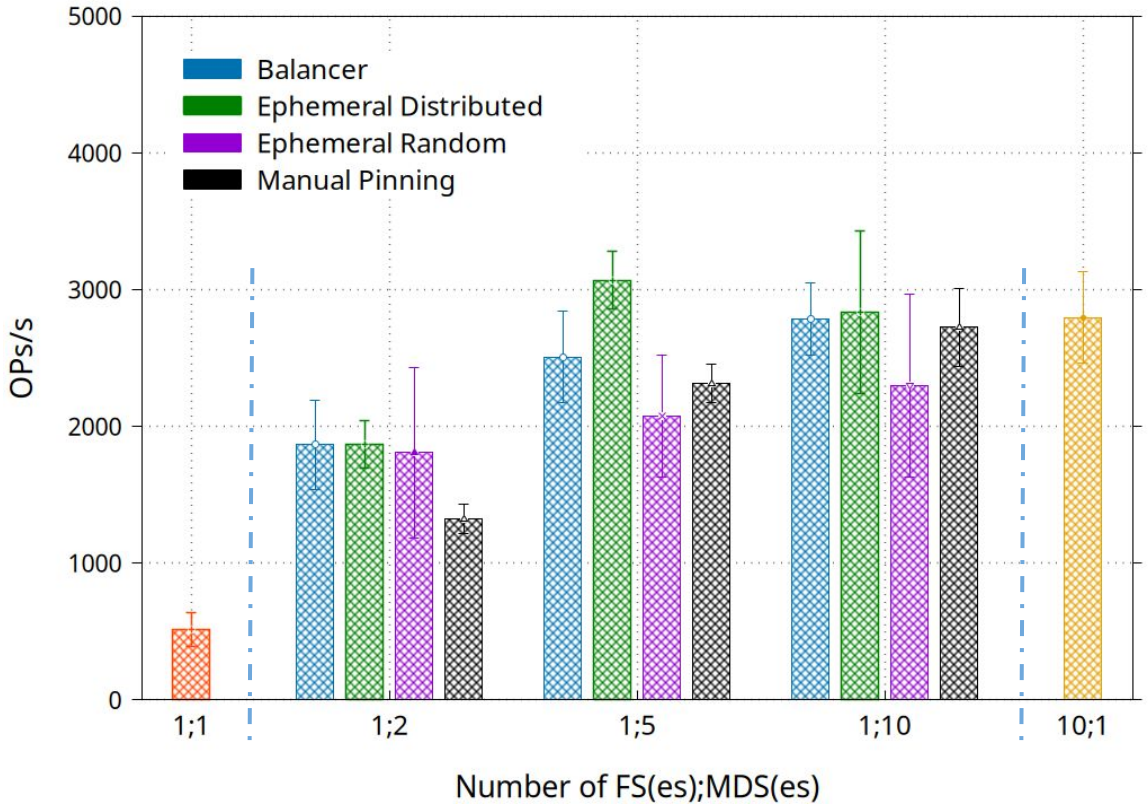


Directory Creation



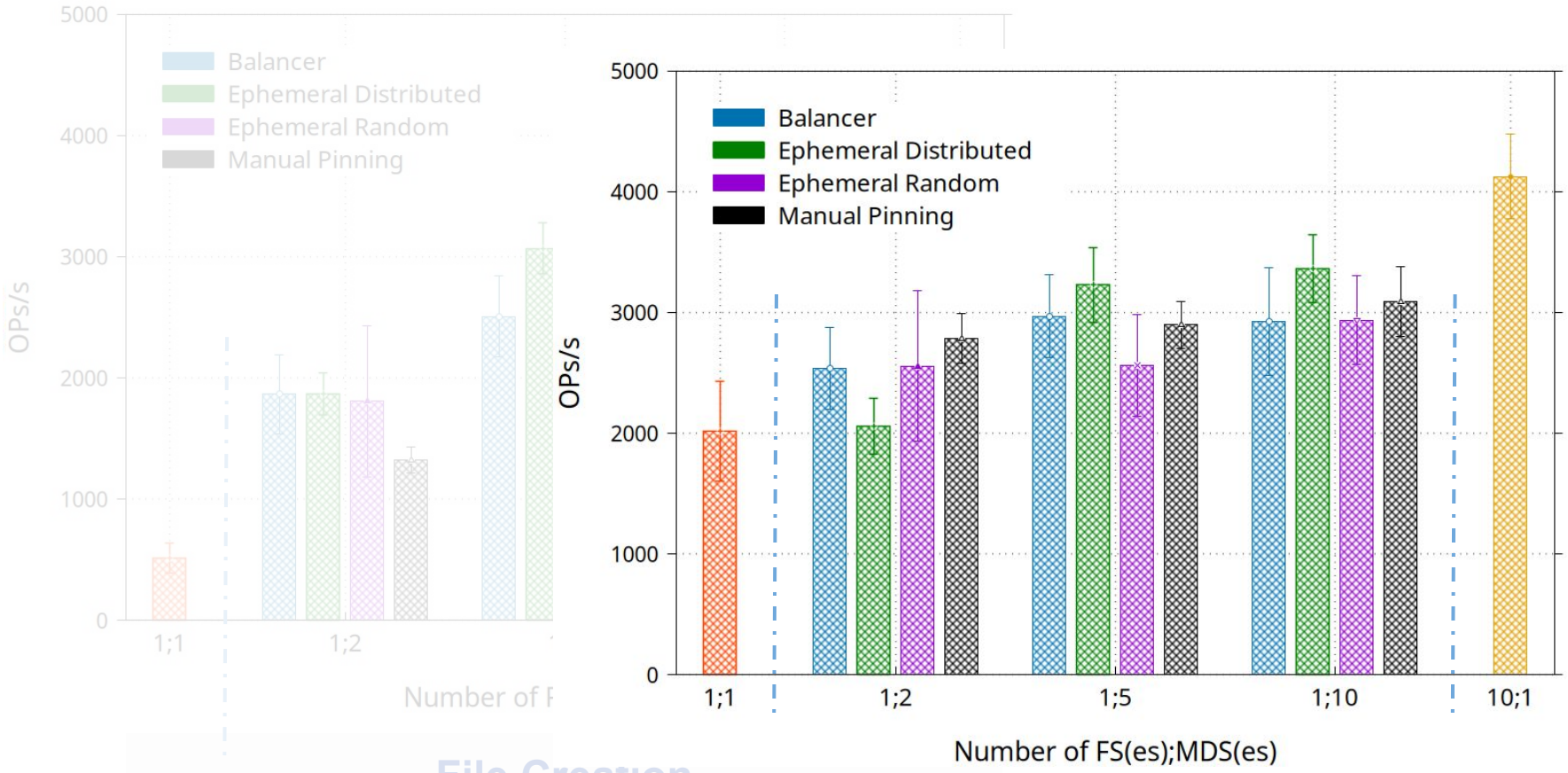
Directory Removal

Metadata Balancing Measurements: Files

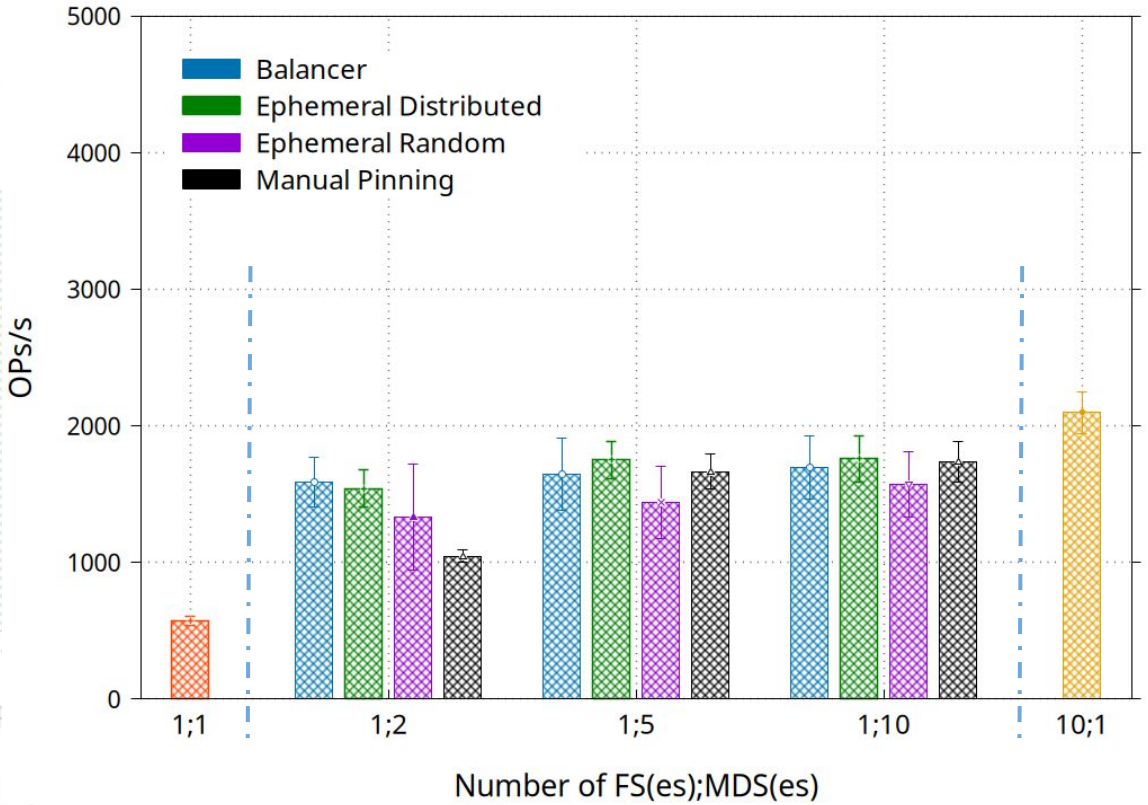


File Creation

Metadata Balancing Measurements: Files



Metadata Balancing Measurements: Files



File Removal

HEALTH_WARNs

1. Automatic Balancer

- Ex-/Im-porting large directory trees makes MDSes very busy
- Checking with wall-clock profiler

```
  cluster:
    id:     15221a48-7392-11f0-9b13-fa163e734f07
    health: HEALTH_WARN
            1 MDSs report slow requests
            2 MDSs behind on trimming

HEALTH_WARN 1 MDSs report slow requests; 2 MDSs behind on trimming
[WRN] MDS_SLOW_REQUEST: 1 MDSs report slow requests
    mds.cephfs.cephp046-c5401080a3.iqgihs(mds.0): 1853 slow requests are blocked > 30 secs
[WRN] MDS_TRIM: 2 MDSs behind on trimming
    mds.cephfs.cephp046-c5401080a3.iqgihs(mds.0): Behind on trimming (586/128) max_segments: 128, num_segments: 586
    mds.cephfs.cephp046-22cf3f4c95.mcowry(mds.1): Behind on trimming (755/128) max_segments: 128, num_segments: 755
```

```
Thread 16 (ms_dispatch) - 100 samples
+ 100.00% __clone3
 + 100.00% start_thread
  + 100.00% AsyncConnection::DelayedDelivery::discard()
   + 100.00% DispatchQueue::entry()
    + 75.00% MDSDaemon::ms_dispatch2(boost::intrusive_ptr<Message> const&)
    |+ 67.00% MDSRankDispatcher::ms_dispatch(boost::intrusive_ptr<Message const> const&)
    ||+ 67.00% MDSRank::_dispatch(boost::intrusive_ptr<Message const> const&, bool)
    || + 55.00% Migrator::dispatch(boost::intrusive_ptr<Message const> const&)
    || |+ 47.00% Migrator::handle_export_dir(boost::intrusive_ptr<MExportDir const> const&)
    || ||+ 46.00% Migrator::decode_import_dir(ceph::buffer::v15_2_0::list::iterator_impl<true>&, int, CDir*, EImportStart*, LogSegment*, std::map<C...
    || |||+ 36.00% Migrator::decode_import_inode(CDentry*, ceph::buffer::v15_2_0::list::iterator_impl<true>&, int, LogSegment*, std::map<CInode*, s...
    || ||||+ 28.00% CInode::decode_import(ceph::buffer::v15_2_0::list::iterator_impl<true>&, LogSegment*)
    || |||||+ 21.00% CInode::_decode_base(ceph::buffer::v15_2_0::list::iterator_impl<true>&)
```

2. `mdtest-easy-delete` similarly produces behind on trimming

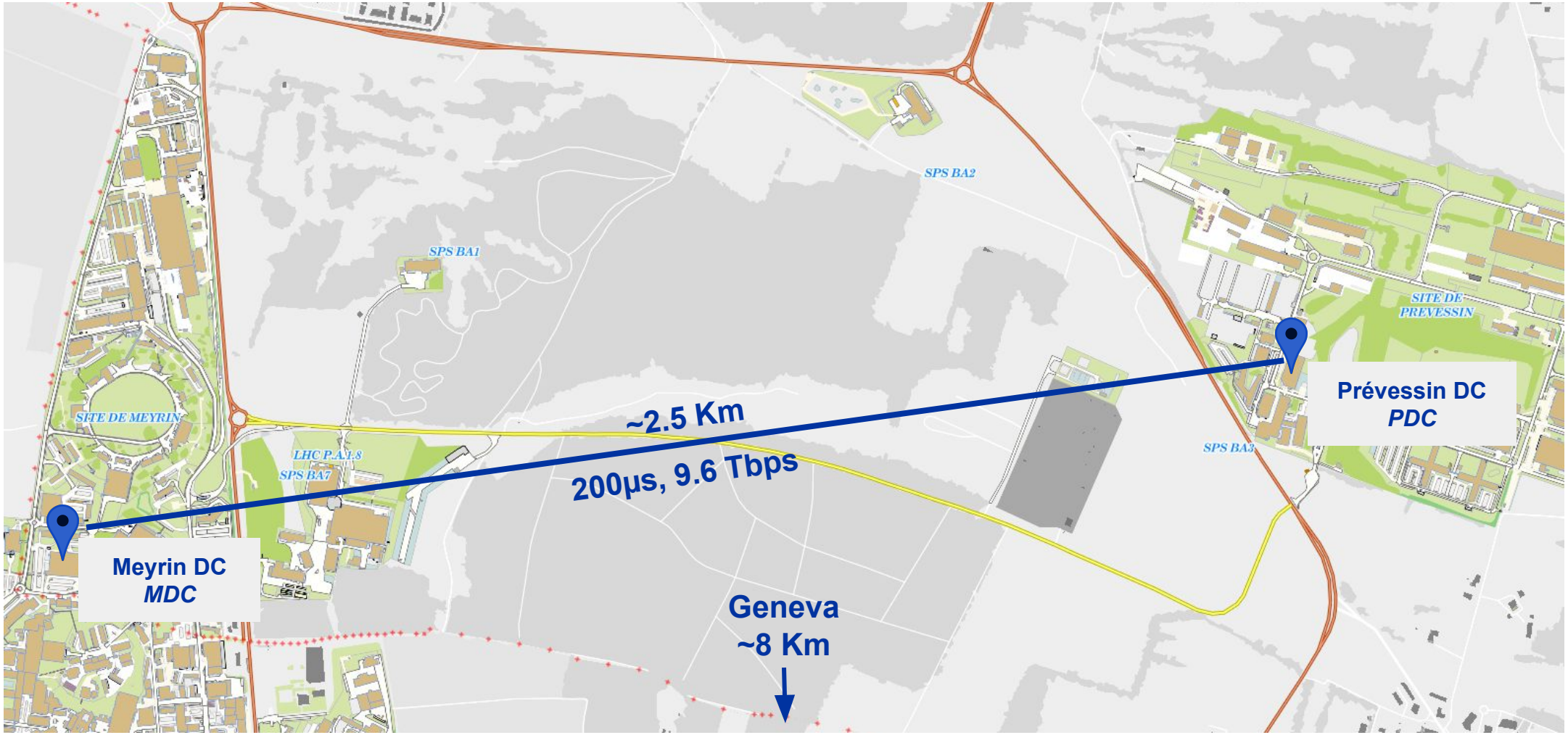
- Particularly visible with low `mds_cache_memory_limit` (e.g., default 4GB)

4

Cross-DC Files and Objects

…and backups on Tape in the future

Recent Addition: New Data Center



CephFS across DCs

Cluster with crush rule maintaining 2 replicas per DC

- Not stretch: Tiebreaker placement is tricky due to power and network constraints
- Stretch mode and exiting-from-stretch have been tested successfully

Leverage data locality whenever possible

- Client population is (much) bigger in one DC
- Crush rule allows to bias read-affinity towards a preferred DC

```
rule cephfsmeta_biasmeyrin_22 {
      id 5
      type replicated
      step take meyrin
      step chooseleaf firstn 2 type pod
      step emit
      step take prevessin
      step chooseleaf firstn 2 type pod
      step emit
}
```

- MDSes do not carry the concept of location
  - Abusing `mds_join_fs` to prefer daemons in one DC
- Clients should read from local OSDs to avoid cross-DC traffic

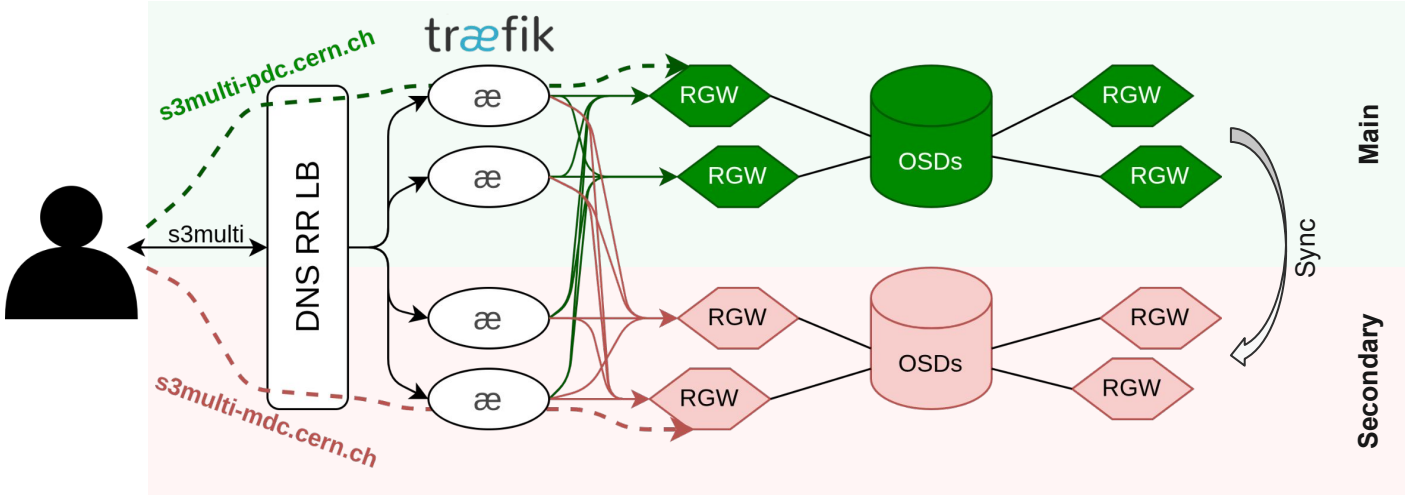
```
mount -t ceph … -o crush_location=datacenter:prevessin,read_from_replica=localize
```

  - Options passed to the kernel client, still reads reach the primary OSD in the other DC :/

S3 Multisite

“Traditional” multisite configuration

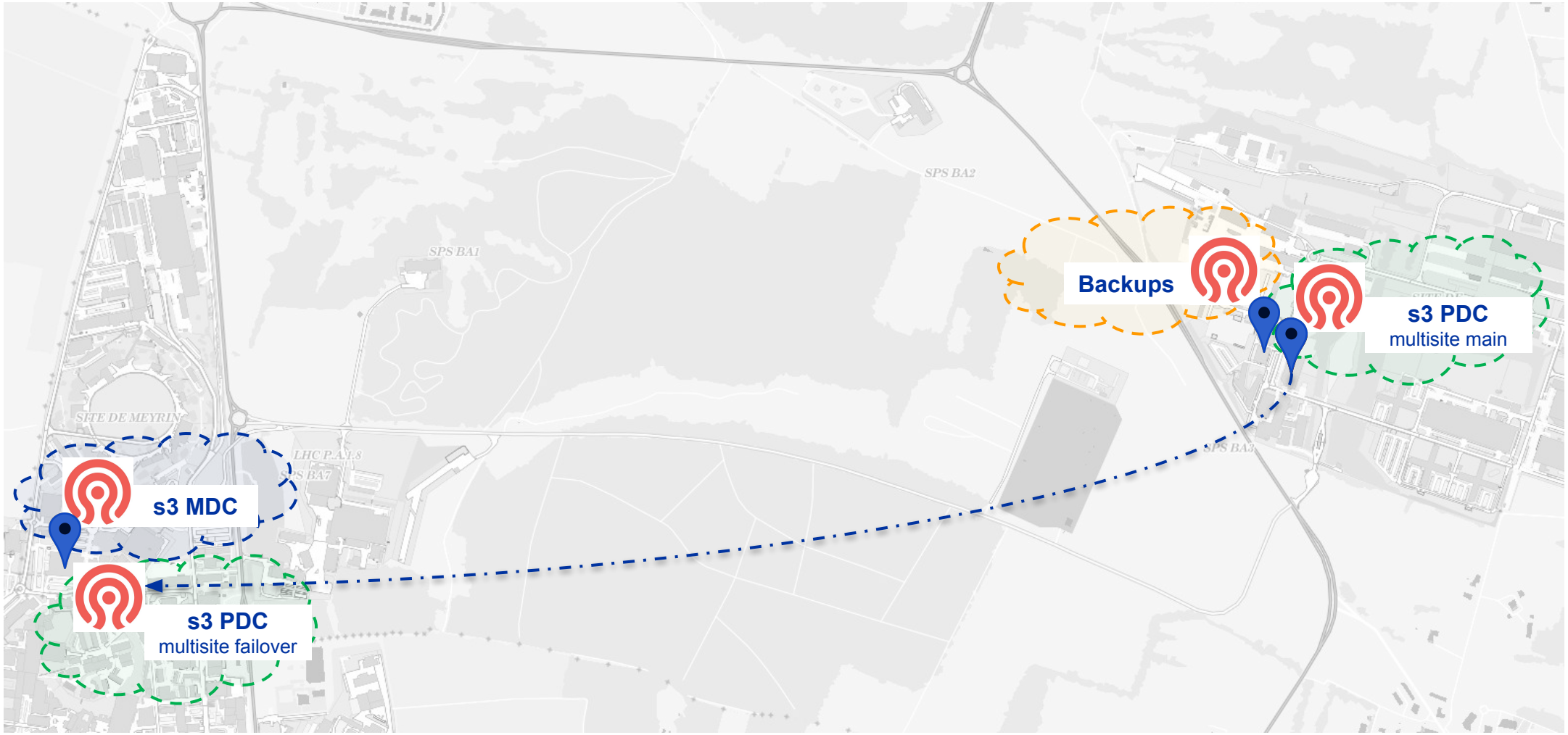
- 2 clusters, 1 zonegroup, 2 zones
  - Full replica main (rw) to secondary (ro)
- Dedicated groups of RadosGWs for:
  - Client traffic: Traefik frontend (L7 routing, TLS termination, health check, failover), and OpenStack Keystone for credentials
  - Sync traffic: Simple RGW with beast



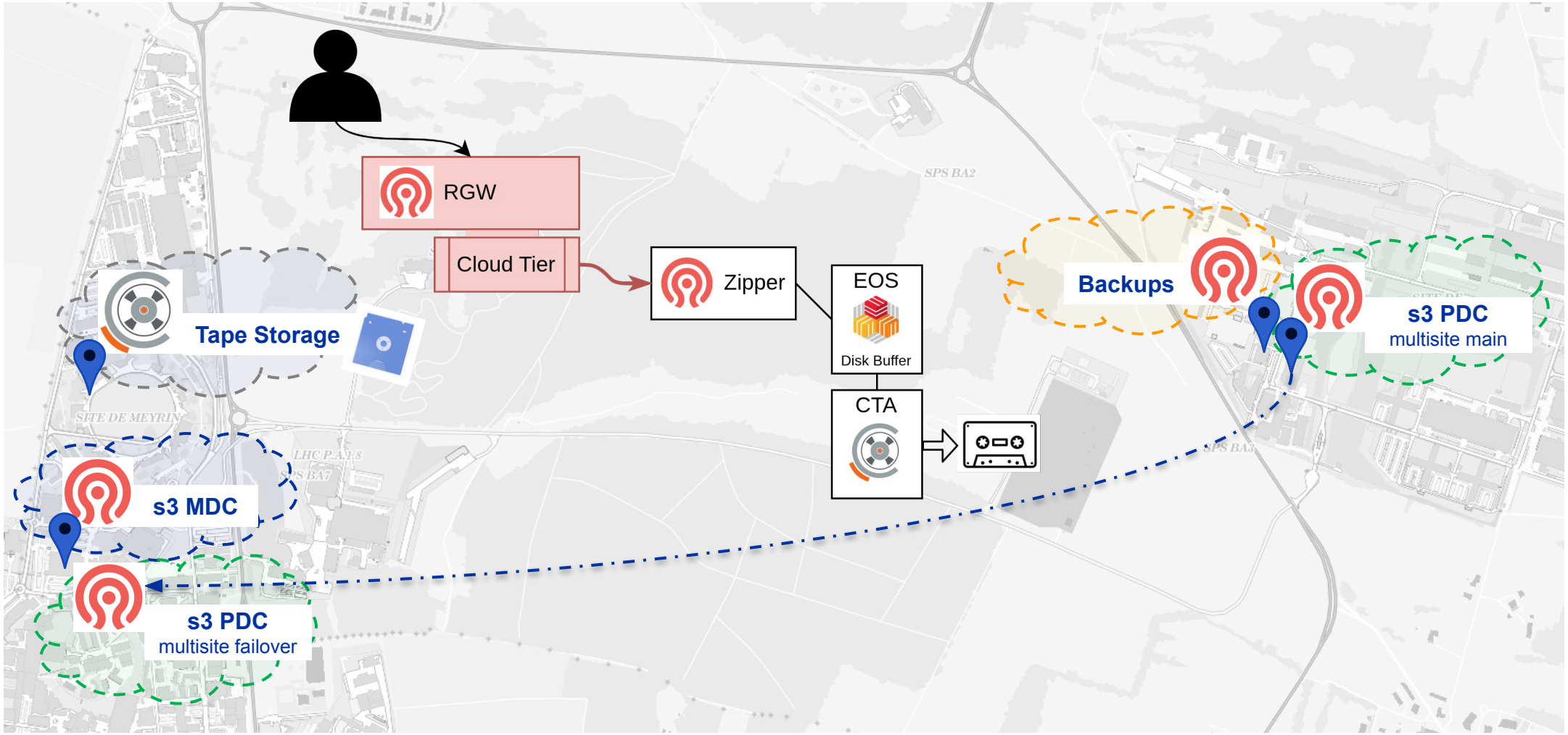
- Recent addition: Each zone is directly addressable through LBs (`s3multi-{pdc,mdc}.cern.ch`)
  - Allows for inferring replication status with new `x-amz-replication-status` header (blog article)

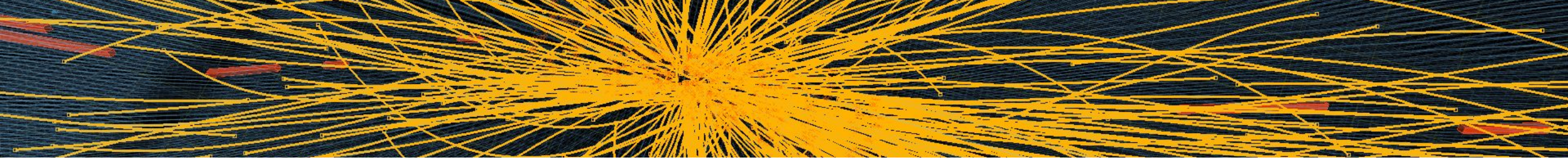
```
# s3cmd --host=s3multi-mdc.cern.ch --host-bucket=s3multi-mdc.cern.ch --debug info s3://enrico/passwd 2>&1 | grep replica
             'x-amz-replication-status': 'REPLICA',
             'x-rgw-replicated-at': 'Thu, 23 Oct 2025 13:33:01 GMT',
             'x-rgw-replicated-from': '13ebcc99-db9e-46e0-81db-90a1556ce13b:enrico:13ebcc99-db9e-46e0-81db-90a1556ce13b.2534364.2'},
```

S3 as Protocol for Backups and Lifecycle Mgmt



S3 as Protocol for Backups and Lifecycle Mgmt





Thank You

Extras



MultiFS Recovery after MON Store Loss

Simulate MON Store recovery using OSDs ⇒ docs

- If successful, MONs + OSDs are back
- Bring 1 MGR back: Re-inject its cephx key (`ceph auth import` helps) or redeploy with `bootstrap-mgr` key

```
$ ceph -s
  cluster:
    id:     6b15f4ce-7973-4f17-b253-66902772c0e5
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum rectest-435fc69142,rectest-0c4d790af5,rectest-0644fef1c2 (age 13m)
    mgr: rectest-0644fef1c2(active, since 10m)
    osd: 144 osds: 143 up (since 2m), 143 in (since 3d)

  data:
    volumes: 3/3 healthy
    pools:   12 pools, 9537 pgs
    objects: 267.33M objects, 48 TiB
    usage:   175 TiB used, 1.6 PiB / 1.8 PiB avail
    pgs:     9520 active+clean
             13   active+clean+scrubbing+deep
             4    active+clean+scrubbing

$ ceph fs ls --format json
[]
```

GOOD

Hmm.. No MDSes?

VERY GOOD

Wait… Where are my FSes gone?!

MultiFS Recovery after MON Store Loss

Bring the MDSes back

- Same as with MGRs: Re-inject cephx keys or redeploy with `bootstrap-mds` key
- They will join the cluster as stand-by as no FS exists

Recreate the File Systems

- Use existing pools (meta + main data) with `--recover` option

```
ceph fs new <fs_name> <metadata_pool> <data_pool> --fscid <fscid> --force --recover


Example:
    $ ceph fs new cephfs cephfs.meta cephfs.data --fscid 1 --force -recover
    $ ceph fs new vancouver vancouver.meta vancouver.data --fscid 2 --force -recover
    $ ceph fs new canada canada.meta canada.data --fscid 3 --force -recover

$ ceph fs ls
name: cephfs, metadata pool: cephfs.meta, data pools: [cephfs.data ]
name: vancouver, metadata pool: vancouver.meta, data pools: [vancouver.data ]
name: canada, metadata pool: canada.meta, data pools: [canada.data ]
```

- Allow standby MDSes to join the FS: `` `ceph fs set cephfs joinable true` ``

MultiFS Recovery after MON Store Loss

What remains to be restored:

- Configuration for any additional data pool
- FS settings stored in the FSmap: `max_mds`, `allow_standby_replay`, `allow_new_snaps`, ...
- All the cephx keys for clients

Purpose of Tests

Characterize behaviour and performance of metadata balancing techniques in a controlled environment, under known testing conditions



- Employed workload does not aspire to represent production; Production is *quite* different…
  - Ability to synthetically reproduce production workload is an open point
- All configuration parameters are left to the defaults, with the exception of
  - `osd_memory_target`=4GB; `osd_mclock_max_capacity_iops_ssd`=25000
  - `mds_cache_memory_limit`=16GB

Test Infrastructure

Server

- 15x bare-metal nodes, Intel Xeon E5-2620 v4 @ 2.10GHz (8 cores, 2 sockets), 128 GB, 10 Gbps
- …each equipped with 15x Micron_5200_MTFDDAK960TDD, SATA 3, 0.96 TB

- AlmaLinux 9.6, 5.14.0-570.28.1.el9_6.x86_64
- Ceph Reef 18.2.7, managed by `cephadm`

Clients

- 10x VMs, 8 cores, 16 GB, 10 Gbps

- AlmaLinux 9.6, 5.14.0-570.28.1.el9_6.x86_64
- CephFS kernel mount, each client mounting filesystem root

Test Instrumentation

Leverage on IO500 benchmarking suite

- `IOR` – Focused on measuring sequential read/write read/write throughput
- `mdtest` – Measures metadata operations performance

Sufficiently flexible

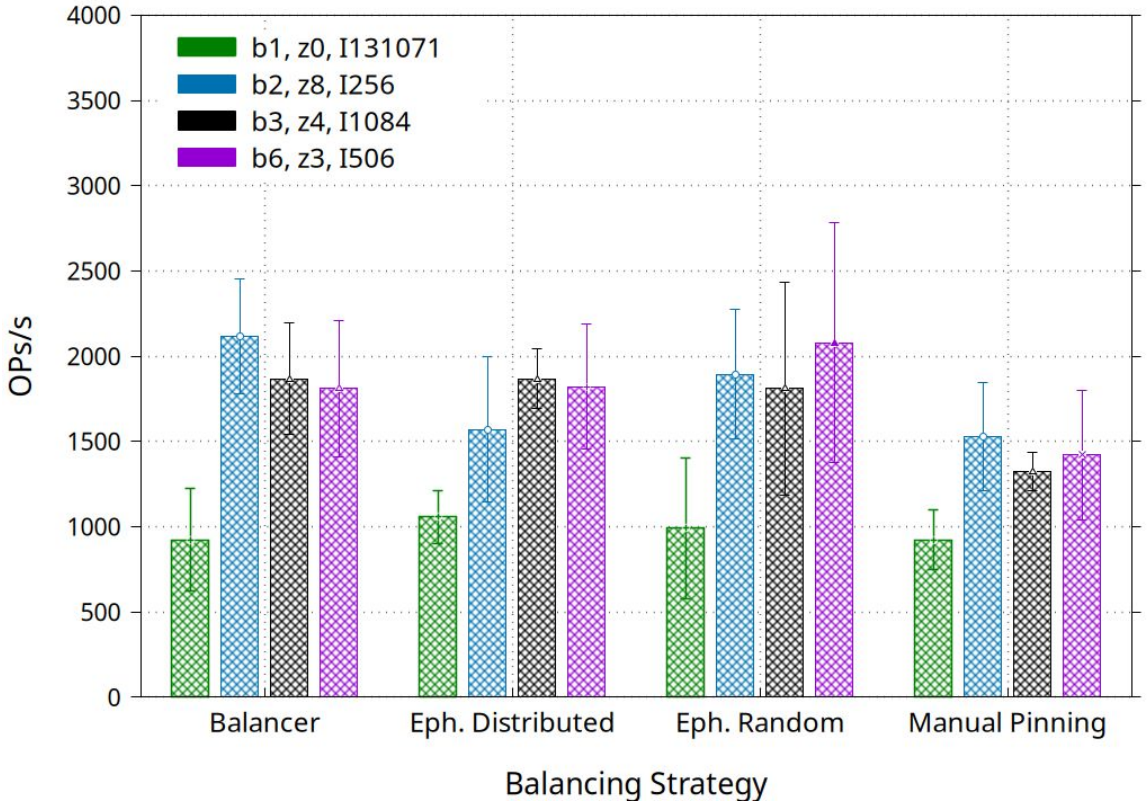
- Create/stat/read/rename/remove files and directories (cherry-pick each test *phase*)
- Depth/branching factor of hierarchical structure; Items (files/dirs) per directory in tree
- Bytes to read/write from/to produced files

Our configuration:

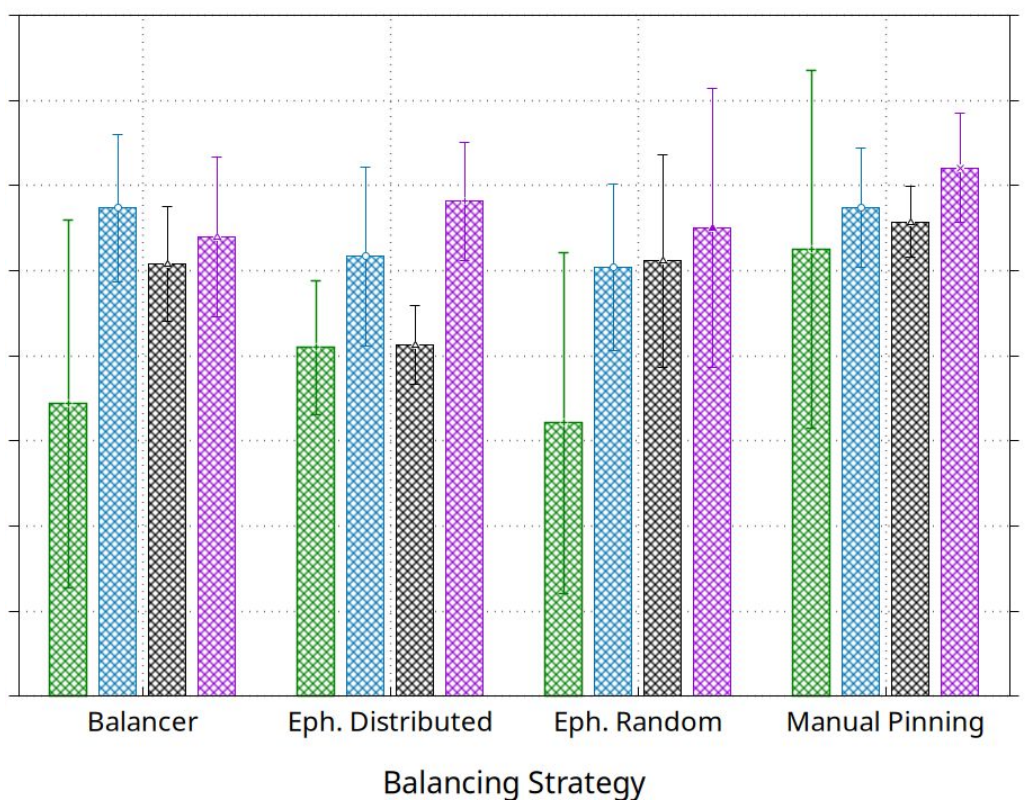
- `mdtest_args='-P -C -i=1 -b=3 -z=4 -I=1084 -w 4096 -e 4096'`
- `--run-cmd-before-phase="echo 3 >/proc/sys/vm/drop_caches; sleep 5"`

- Resulting in each client (10 in total) producing 250k items (~50% files / 50% dirs)

Metadata Balancing: Impact of Directory tree layout

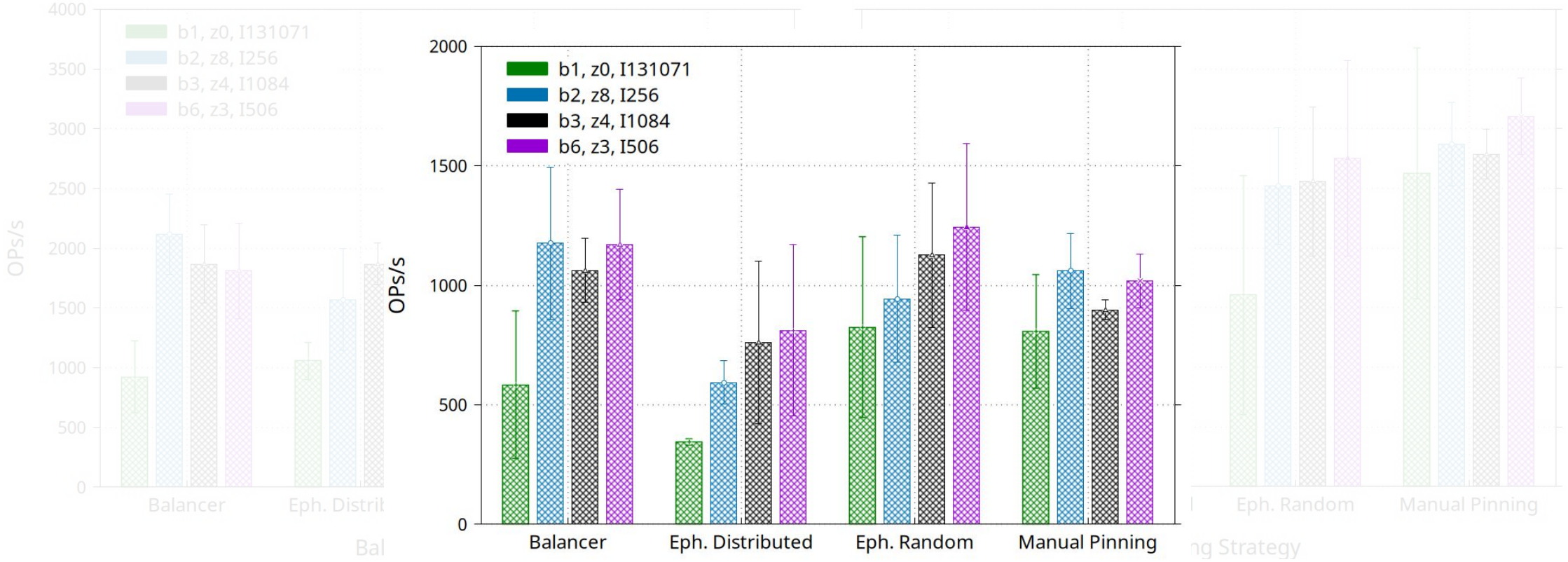


File Creation



File Stat

Metadata Balancing: Impact of Directory tree layout



Directory Removal

Ceph Science Users Group

An informal open call for community members primarily from hpc/htc/research environments, though anyone is welcome!



Topics discussed:

- Updates, outages, features, maintenance, …
- Open floor to discuss anything else about Ceph
- Each meeting features a short presentation – Contributions are welcome!

When and where:

- Every 2 months, last Tuesday of the month, 14:00 UTC
- Check on ceph.io/.../meetups/ for schedule and local time

Find previous meetings at:

- Notes: pad.ceph.com
- Video recordings: YouTube