


# Ceph Days Silicon Valley 2025

## Supporting 3 Availability Zones Stretch Cluster

Kamoltat (Junior) Sirivadhna

# Agenda

- Introduction
- Problems with traditional stretch cluster setup
- Stretch Mode (2 Availability Zones)
- Expanding to 3 Availability Zones
- Current Limitations & Progress
- Q&A

**Github:** https://github.com/kamoltat

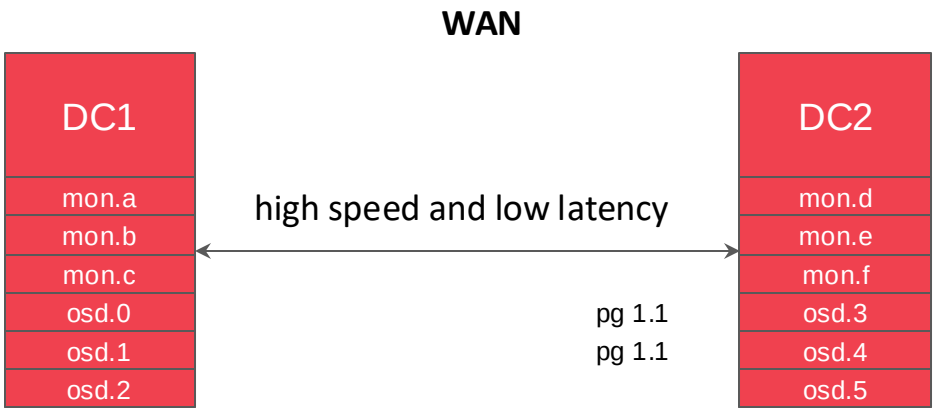
## Kamoltat (Junior) Sirivadhna

- **Role:** Software Engineer
- **Team:** RADOS / Teuthology
- **Experiences:**
  - 5+ Years of Ceph Contribution
- **Key Contributions:**
  - Stretch Cluster / Stretch Mode
  - PG Autoscaler
  - Teuthology

# Stretch Cluster

Stretch Cluster **should**:

- Withstand netsplit
- Can still serve I/O with a single datacenter




```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f}

pg 1.1 = {0,1,3,4}
```

# Monitor Election Strategy: Connectivity

- Each monitor maintains connection scores of its peers
- Scores are shared between all the monitors
- Monitor with highest score is elected leader, if scores are tied, the monitor with the lowest rank wins.

# mon.a POV

mon.a vs mon.b in an election …

mon.a score = 0.94 + 0.98 / 2 = 0.96
mon.b score = 0.53 + 0.59 / 2 = 0.56
0.96 > 0.56
**mon.a wins!**

**Monitor Scoring Report**

- mon.a scores: {mon.b: **0.53**, mon.c: **0.83}**
- mon.b scores: {mon.a: **0.94**, mon.c: **0.74}**
- mon.c scores: {mon.a: **0.98**, mon.b: **0.59}**

# Stretch Cluster: Problem with Traditional Setup

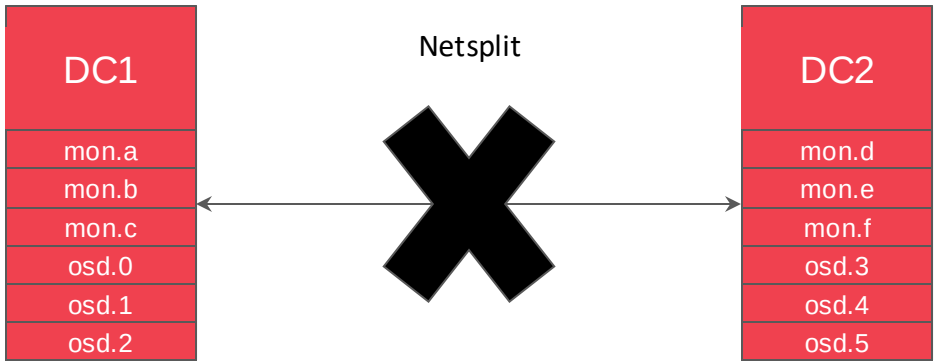
- Netsplit
- Peering rule

# Stretch Cluster: Netsplit Problem (MON election stuck)

mon.a: {mon.a, mon.b, mon.c} votes
mon.d: {mon.d, mon.e, mon.f) votes

No one wins because you need number of votes > half of quorum

Election Cycles Forever!




Cluster becomes inaccessible!

```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f}

pg 1.1 = {0,1,3,4}
```

# Stretch Cluster: Problem with Traditional Setup

- Netsplit
- Peering rule

# Peering Rule:

- A set of conditions when choosing OSDs to be in the acting_set of a PG.
- This is also crucial in determining whether a PG should go active (accept I/O) or not.

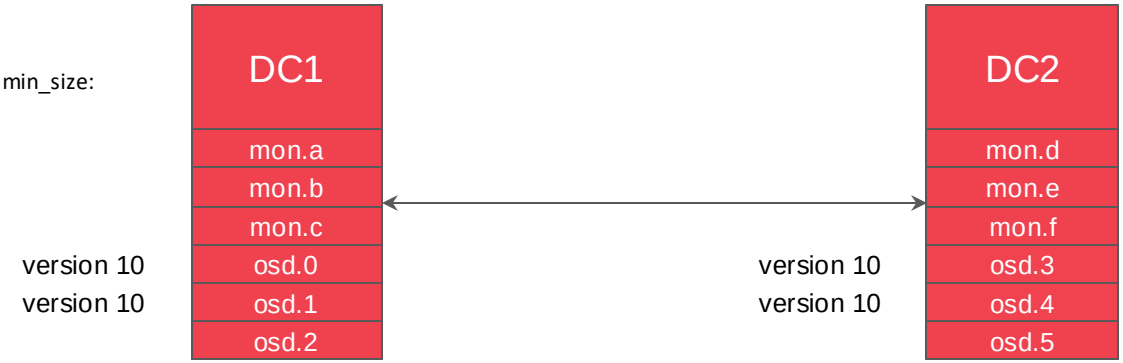
**Trivial rule:**

```
if acting_set size >= min_size:
   PG go active!
```

# Stretch Cluster: Peering Problem

Peering rule:

if acting_set size >= min_size:
 PG go active!

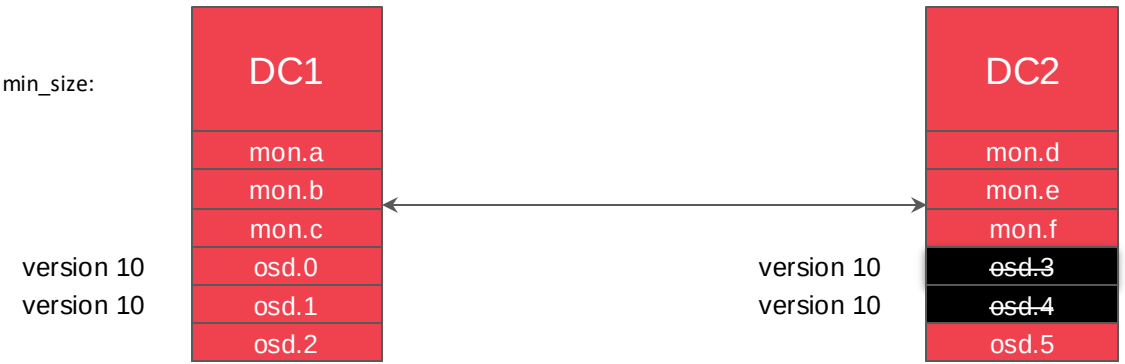



```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f}

pg 1.1 = {0,1,3,4}
```

# Stretch Cluster: Peering Problem

Peering rule:

if acting_set size >= min_size:
PG go active!

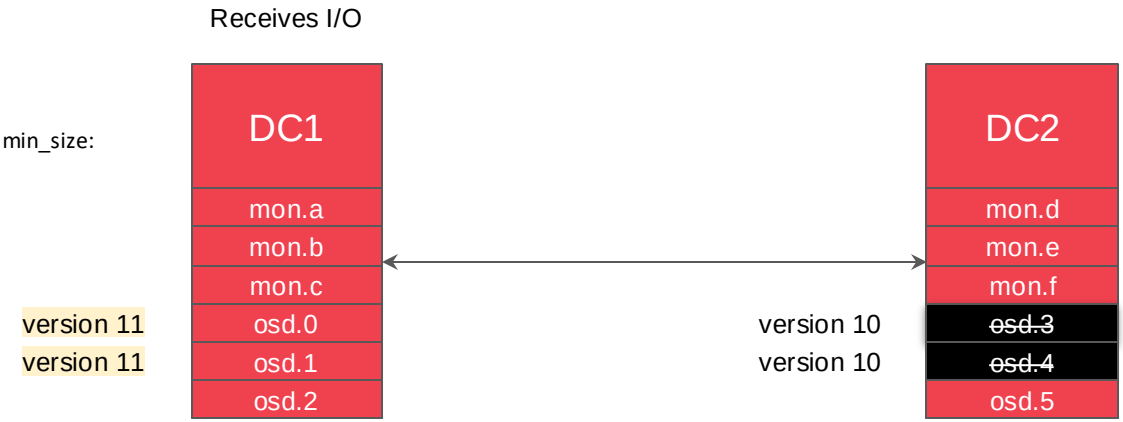



```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f}

pg 1.1 = {0,1}
```

# Stretch Cluster: Peering Problem

Peering rule:

if acting_set size >= min_size:
PG go active!

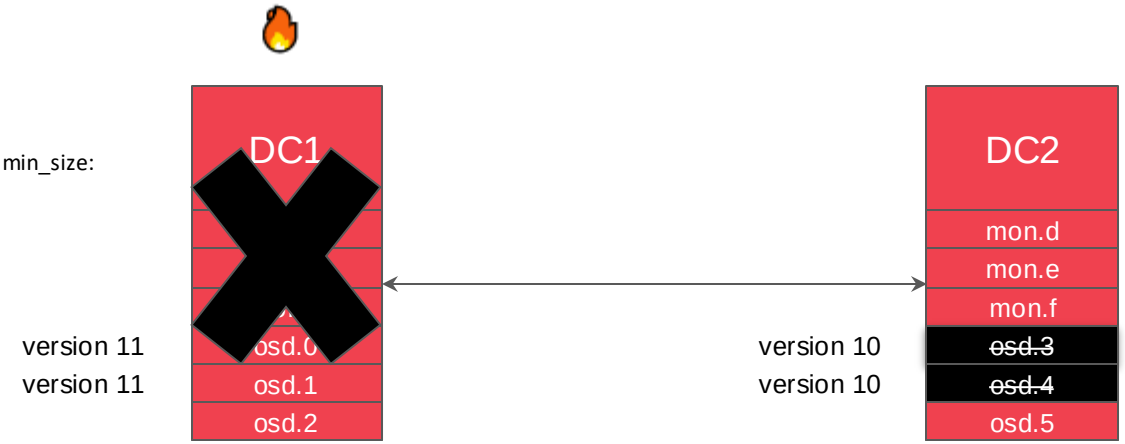



```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f}

pg 1.1 = {0,1}
```

# Stretch Cluster: Peering Problem

Peering rule:

if acting_set size >= min_size:
PG go active!



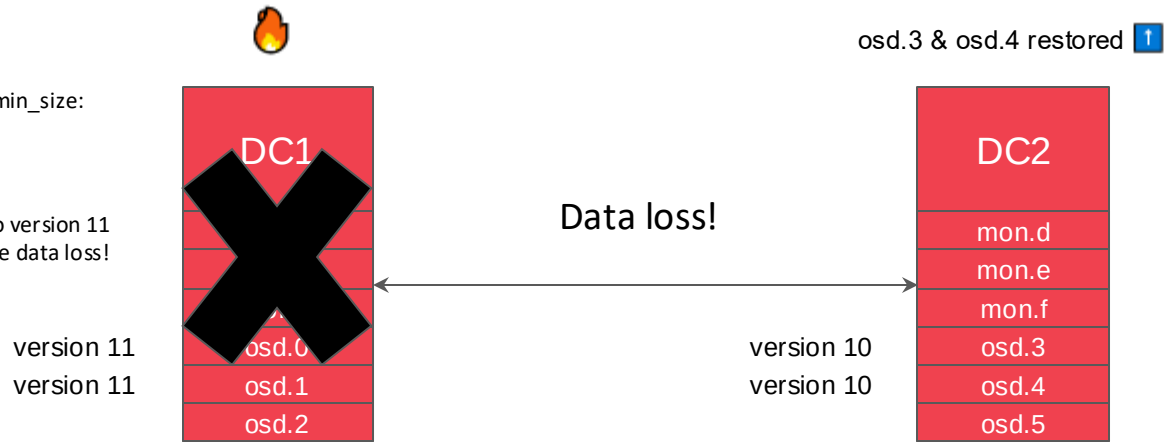

```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f}

pg 1.1 = {}
```

# Stretch Cluster: Peering Problem

Peering rule:

if acting_set size >= min_size:
PG go active!

osd.3 and osd.4
can never get back to version 11
which means we have data loss!



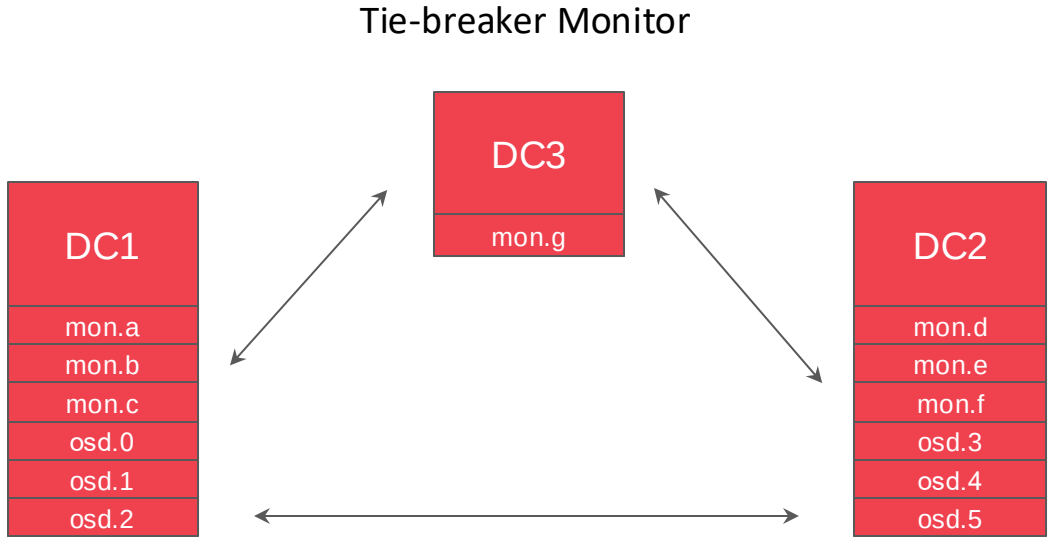

```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f}

pg 1.1 = {3,4}
```

# Stretch Cluster: Problem with Traditional Setup

- Netsplit
- Peering rule

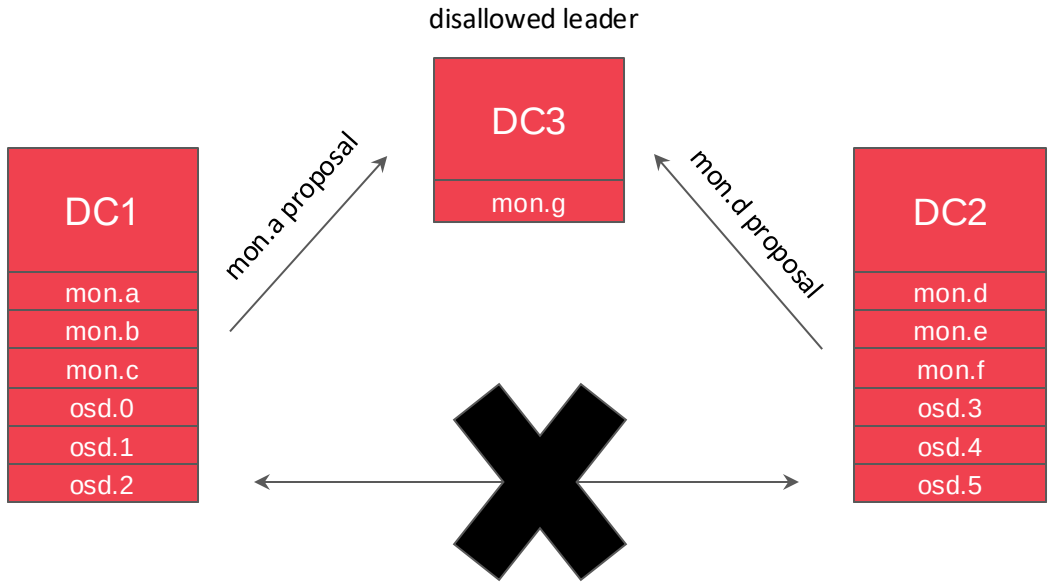
**Stretch Mode is the solution!**

# Stretch Mode




```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f,g}

pg 1.1 = {0,1,3,4}
```

# Stretch Mode: Netsplit



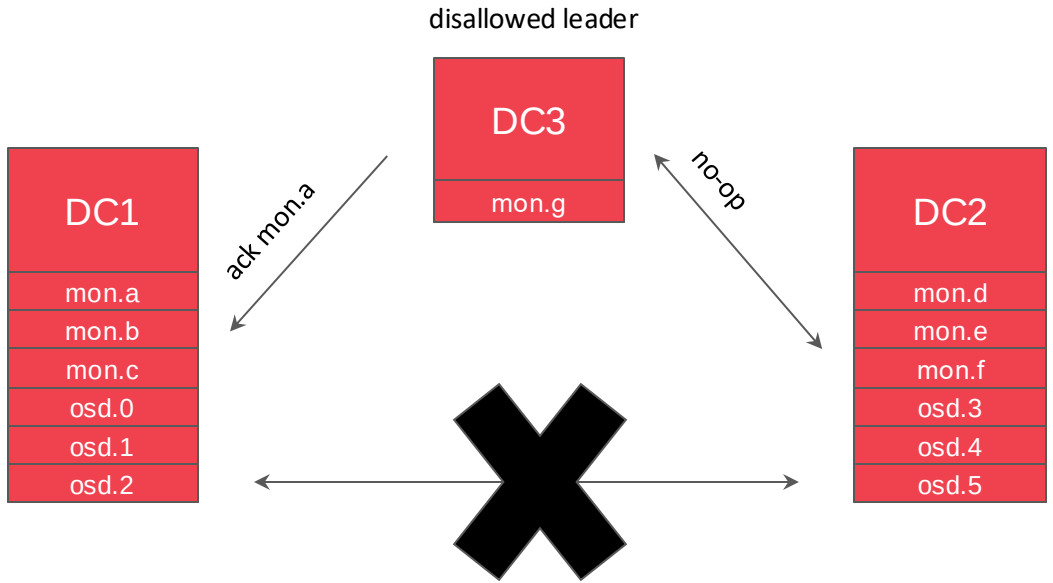

```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum: unknow (electing)

pg 1.1 = {0,1,3,4}
```

# Stretch Mode: Netsplit

mon.a: {a,b,c,g} votes
mon.d: {d,e,f} votes

mon.a # of votes > half of quorum

mon.a wins the election!




```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,g}

pg 1.1 = {0,1,3,4}
```

# Stretch Mode: Netsplit

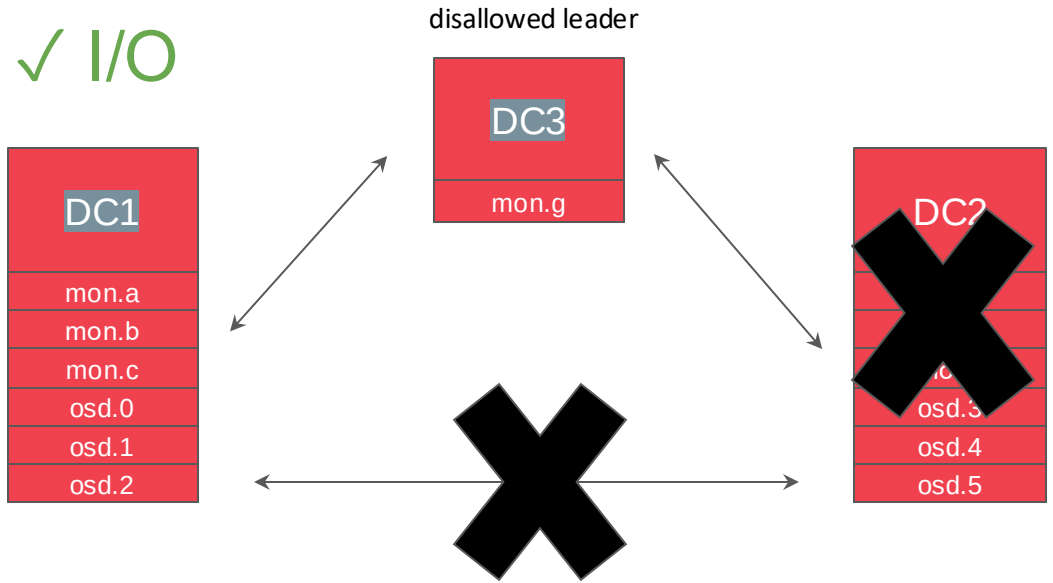
mon.a: {a,b,c,g} votes
mon.d: {d,e,f} votes

mon.a # of votes > half of quorum

mon.a wins the election!

DC2 is kept in the dark

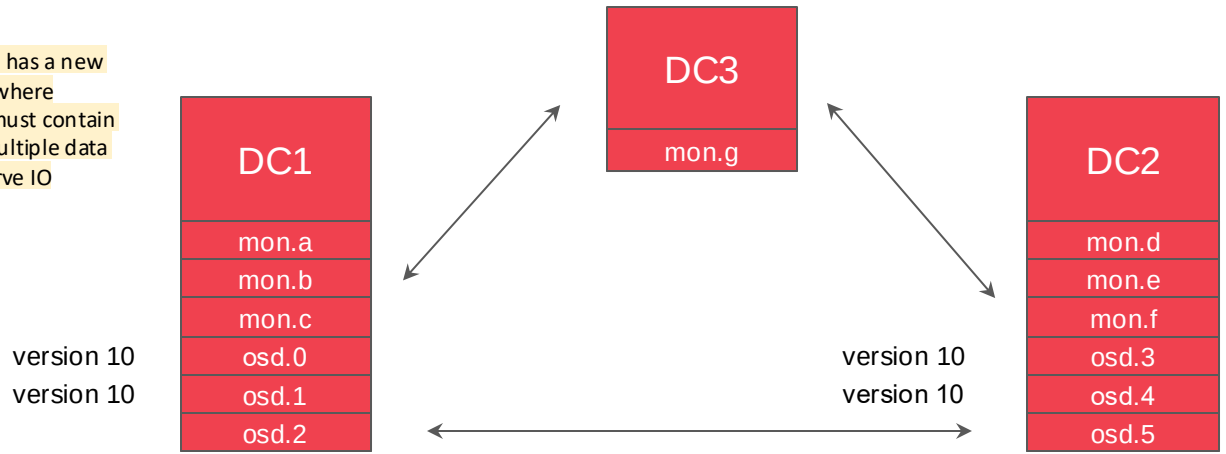
DC1 and DC3 remains operational in degraded mode




```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 2
min_size = 1

quorum {a,b,c,g}

pg 1.1 = {0,1}
```

# Stretch Mode: Stretch peering rule

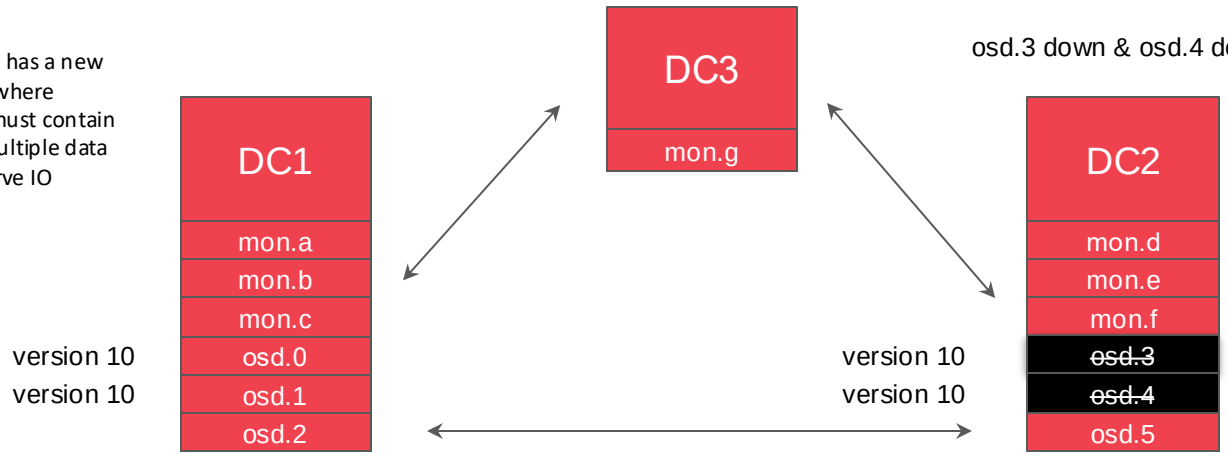
Stretch mode has a new peering rule where "acting set" must contain OSDs from multiple data centers to serve IO




```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f,g}

pg 1.1 = {0,1,3,4}
```

# Stretch Mode: Stretch peering rule

Stretch mode has a new peering rule where "acting set" must contain OSDs from multiple data centers to serve IO



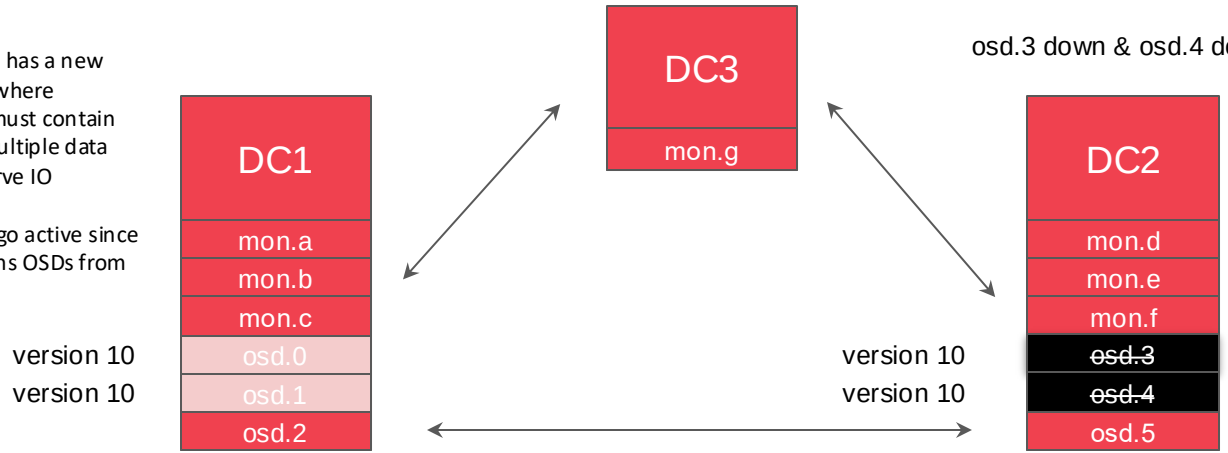

osd.3 down & osd.4 down

```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f,g}

pg 1.1 = {0,1}
```

# Stretch Mode: Stretch peering rule

Stretch mode has a new peering rule where "acting set" must contain OSDs from multiple data centers to serve IO

pg.1.1 won't go active since it only contains OSDs from DC1



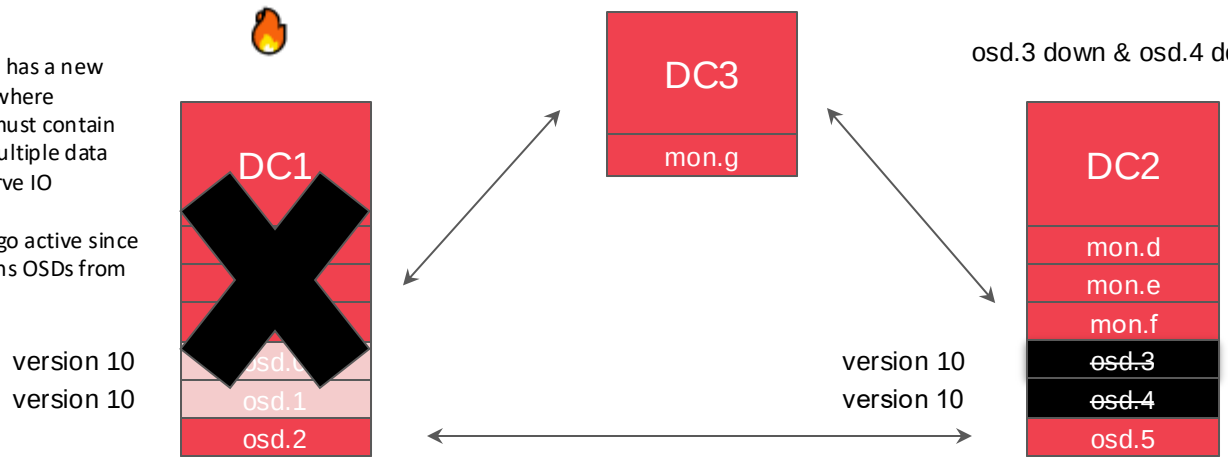

osd.3 down & osd.4 down

```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {a,b,c,d,e,f,g}

pg 1.1 = {0,1}
```

# Stretch Mode: Stretch peering rule

Stretch mode has a new peering rule where "acting set" must contain OSDs from multiple data centers to serve IO

pg.1.1 won't go active since it only contains OSDs from DC1



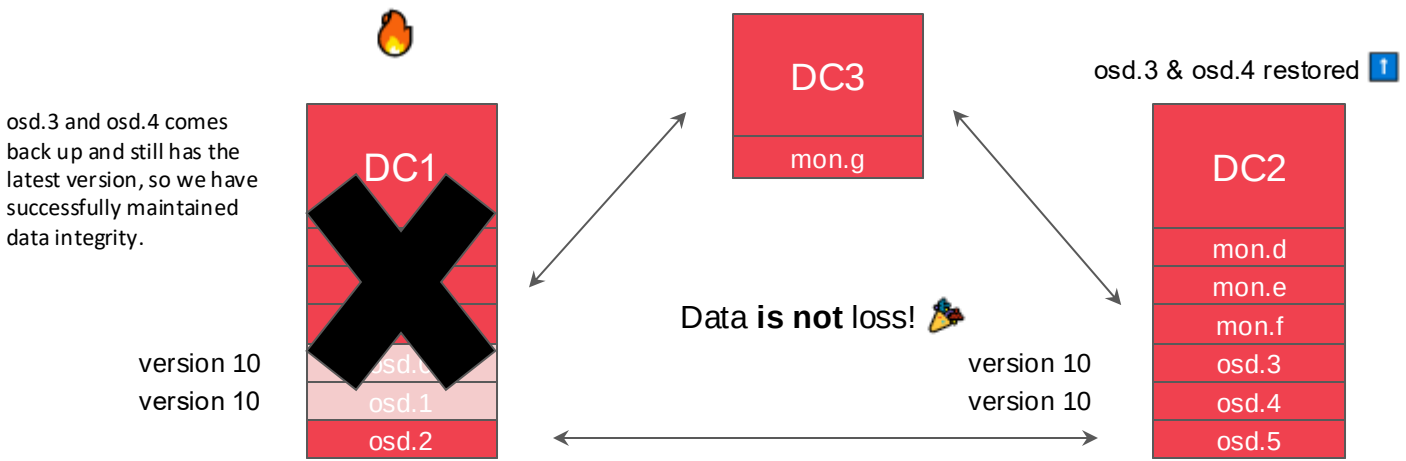

```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}
```

size = 4
min_size = 2

quorum {d,e,f,g}

pg 1.1 = {}

# Stretch Mode: Stretch peering rule

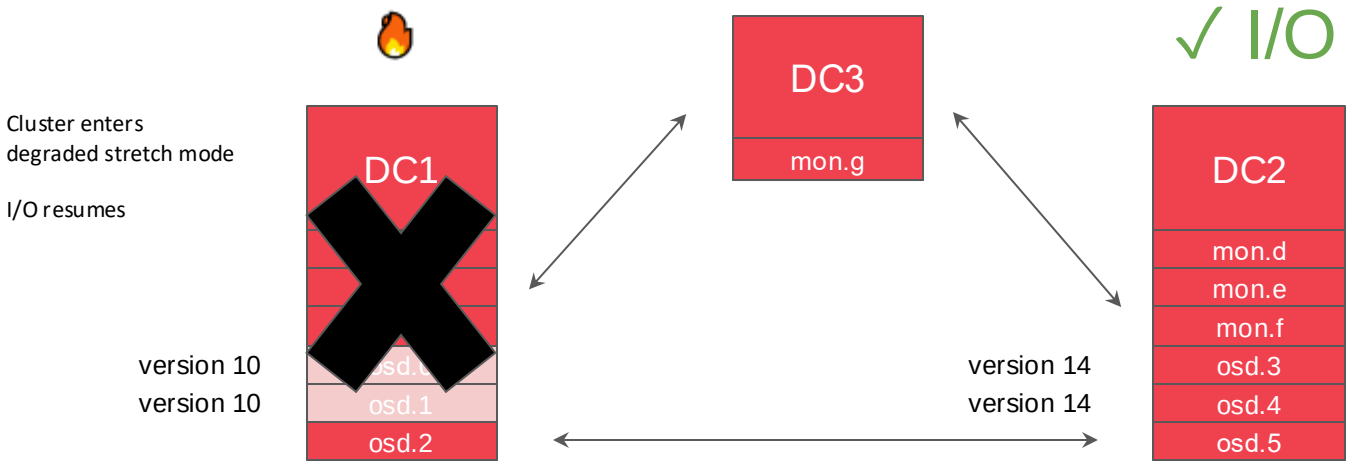



```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 2

quorum {d,e,f,g}

pg 1.1 = {3,4}
```

## Stretch Mode: degraded mode

- Triggers when all monitors in a datacenter fails
- Reduces the min_size to 1
- Peering rule allows surviving site to serve I/O

# Stretch Mode: Stretch peering rule in degraded mode:




```
rule stretch_rule {
    id 2
    type replicated
    step take default
    step choose firstn 2 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}

size = 4
min_size = 1

quorum {d,e,f,g}

pg 1.1 = {3,4}
```



# Stretch Cluster: 3 Availability Zones

- Stretch Mode only works with 2 sites
  - Cluster wide configuration
- How do we expand to 3 sites?




# Stretch Cluster: 3 Availability Zones Peering rule

- Generalize the stretch peering rule! Allowing the user to config the rule.

**Introducing a new Ceph Command:**

*ceph osd pool stretch set*

*<pool-name> <peering_crush_bucket_count> <peering_crush_bucket_target>*

*<peering_crush_bucket_barrier> <crush_rule> <size> <min_size>*

# Stretch Cluster: 3 Availability Zones Peering rule

**peering_crush_bucket_count:**

The minimum number of distinct bucket (data center) an acting set is expected to have in order for a PG to go active.

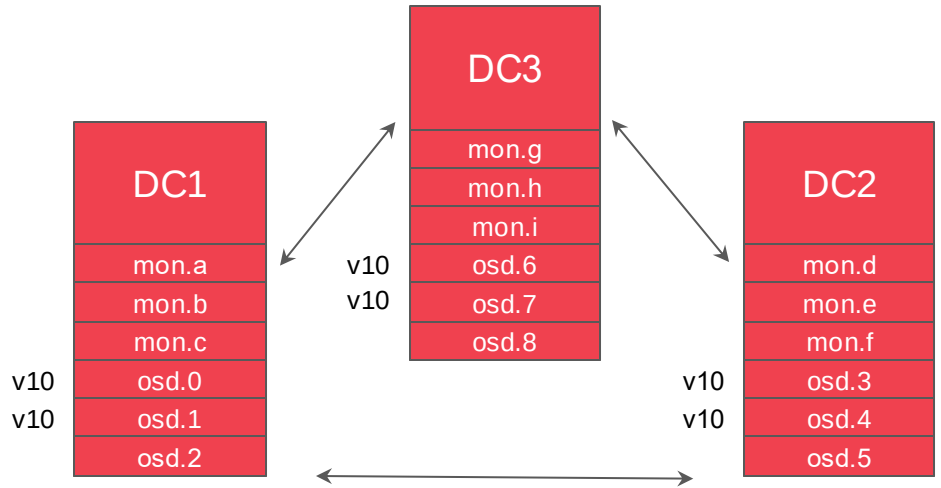
**peering_crush_bucket_target:**

A value used in conjunction with `size` to determine the maximum number of OSDs from the same bucket allowed to be chosen for the acting set (bucket_max).

**peering_crush_bucket_barrier:**

The type of bucket a pool is stretched across, e.g., datacenter and zone.

# Stretch Cluster: 3 Availability Zones

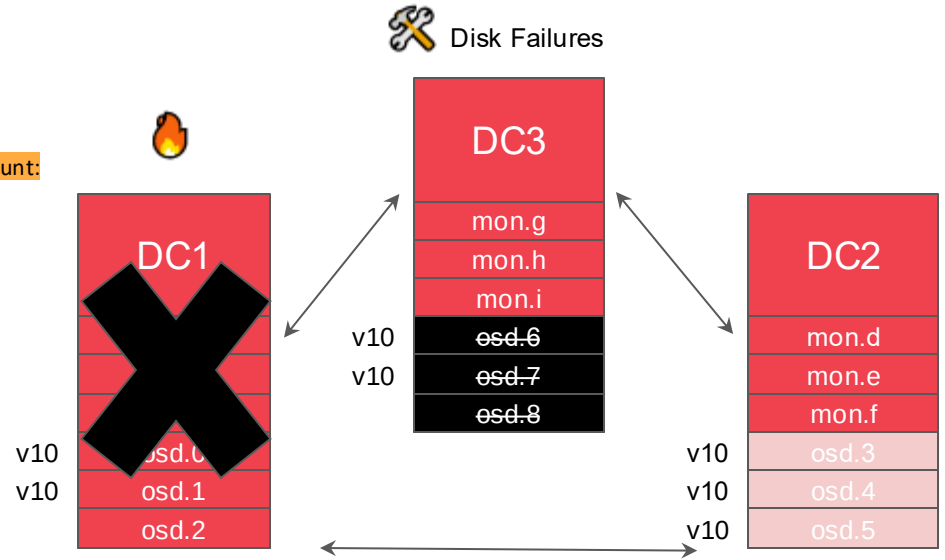



```
rule 3az_rule {
        id 1
        type replicated
        step take default
        step choose firstn 3 type datacenter
        step chooseleaf firstn 2 type host
        step emit
 }

peering_crush_bucket_count = 2
peering_crush_bucket_target = 3
peering_crush_bucket_barrier = datacenter

size = 6
min_size = 3

quorum {a,b,c,d,e,f,g,i}

pg 1.1 = {0,1,3,4,6,7}
```

# Stretch Cluster: 3 Availability Zones

if number of distinct buckets in acting_set < peering_crush_bucket_count: PG cannot go active!

Since osd.3, osd.4 and osd.5 belong to DC2, the PG cannot go active!




```
rule 3az_rule {
        id 1
        type replicated
        step take default
        step choose firstn 3 type datacenter
        step chooseleaf firstn 2 type host
        step emit
}
```

peering_crush_bucket_count = 2
peering_crush_bucket_target = 3
peering_crush_bucket_barrier = datacenter

size = 6
min_size = 3

quorum {g, h, i, d,e,f}

pg 1.1 = {3,4,5} (5 gets added due to rebalance)

# Stretch Cluster: 3 Availability Zones Peering rule

**peering_crush_bucket_count:**

The minimum number of distinct bucket (data center) an acting set is expected to have in order for a PG to go active.

**peering_crush_bucket_target:**

A value used in conjunction with `size` to determine the maximum number of OSDs from the same bucket allowed to be chosen for the acting set (bucket_max).

**peering_crush_bucket_barrier:**

The type of bucket a pool is stretched across, e.g., datacenter and zone.

## peering_crush_bucket_target

If we ended up losing a datacenter, we don't want to shove 3 datacenters worth of replica into a single site, so we calculate **BUCKET_MAX**, which limits the number of OSDs we pull from any given bucket even if it leaves us undersized

**BUCKET_MAX = SIZE / PEERING_CRUSH_BUCKET_TARGET**

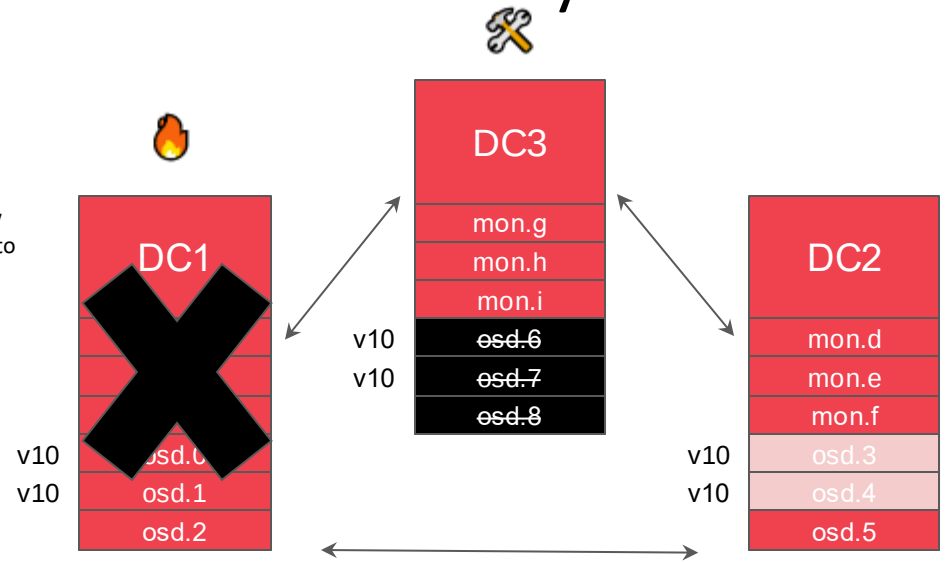
**BUCKET_MAX = 6 / 3 = 2**

**This means:**

**If I have 3 failure domains of size 6, What's the maximum number OSDs I can choose from each site … the answer is 2!**

# Stretch Cluster: 3 Availability Zones

pg.1.1 will remain undersize since we already have two OSDs from DC2, bucket_max = 2 won't allow more than two OSDs from each site into the acting set!




```
rule 3az_rule {
        id 1
        type replicated
        step take default
        step choose firstn 3 type datacenter
        step chooseleaf firstn 2 type host
        step emit
}

peering_crush_bucket_count = 2
peering_crush_bucket_target = 3
peering_crush_bucket_barrier = datacenter
bucket_max = 2

size = 6
min_size = 3

quorum {d,e,f,g}

pg 1.1 = {3,4}
```

# Stretch Cluster: 3 Availability Zones

- Utilize stretch peering rule ✓
- Withstanding Netsplit 🛠
  - Netsplit Detection (PR currently under review)
    - https://github.com/ceph/ceph/pull/59248
  - Picking the surviving site 🛠

# Stretch Cluster: 3 Availability Zones Split Brain Problem

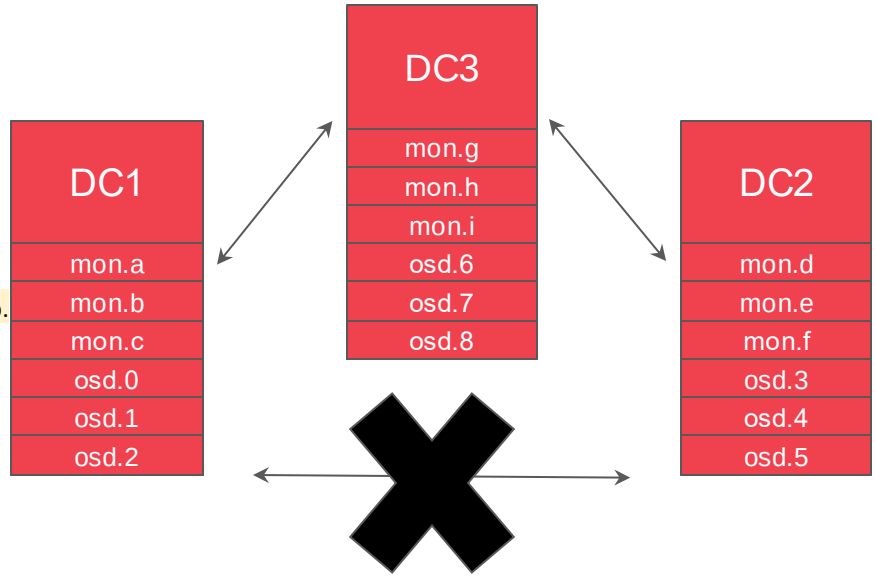
osd.3 reports that osd.0 is down
osd.4 reports that osd.0 is down

leader monitor marks osd.0 as down.

osd.6 reports that osd.0 is up
osd.7 reports the osd.0 is up

leader monitor then marks osd.0 back up.

This cycles continues endlessly and the PG gets stuck in peering because of this. Meaning, PG cannot receive I/O until netsplit is resolved …




```
rule 3az_rule {
        id 1
        type replicated
        step take default
        step choose firstn 3 type datacenter
        step chooseleaf firstn 2 type host
        step emit
}
```

size = 6
min_size = 3

quorum {a,b,c,d,e,f,g,i}

pg 1.1 = {0,1,3,4,6,7}

# Challenges with 3 Availability Zones Netsplit:

- Stretch-mode (two-sites) deals implicitly by having the tie-breaker monitor … Why can’t we make one of the sites in 3 AZ a tie-breaker?
- Designating one of 3 data zones as a tie-breaker reduces the zone’s importance, defeating the purpose of having three availability zones.
- Solution to the split brain problem:
  - Detect Netsplit
  - Make a decision which zones we choose to survive



# Netsplit Detection (Feature under review)

https://github.com/ceph/ceph/pull/59248

- Build a **network graph structure** using **connection scores** from the monitors
- Maps monitor disconnections to relevant CRUSH topology levels.
- Aggregates individual disconnections into location-level reports when appropriate
- Falls back to individual monitor-level reporting for partial disconnections

The implementation produces health warnings in **ceph -s**:

1. For complete location netsplits:

**MON_NETSPLIT: "Netsplit detected between dc1 and dc2"**

2. For individual monitor disconnections, e.g., partial disconnection (not whole data center):

**MON_NETSPLIT:"Netsplit detected between mon.a and mon.d"**

# Pick the surviving sites (POC):

## Netsplit Decision Heuristics

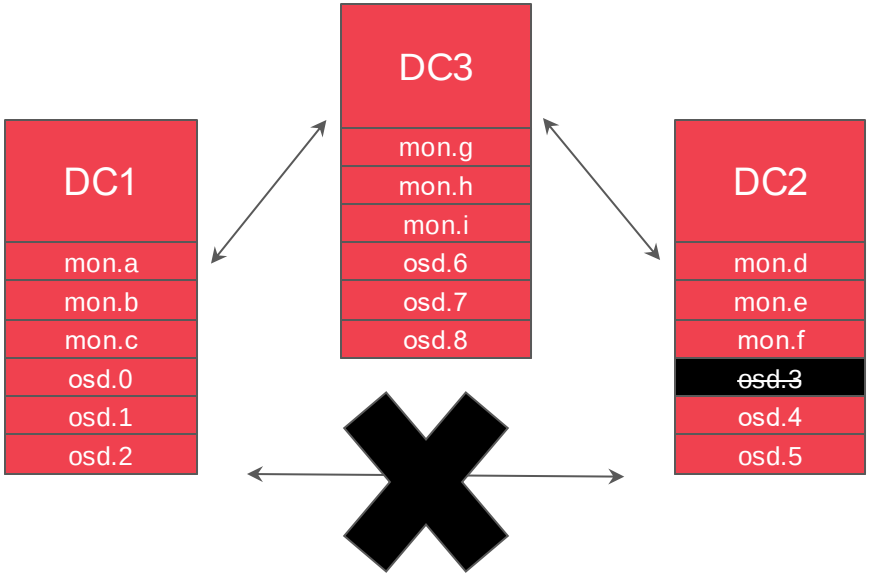
### Primary Factors (In Priority Order):

1. **OSDs (Up) Count** – Prioritizes data availability, minimizes recovery.
2. **Monitor Count** – Ensures control plane stability and quorum resilience.
3. **Connection Scores** – Prefers sites with more reliable communications.

### Tiebreaker:

- **Lowest Combined Monitor Rank** – Ensures a **deterministic, repeatable decision** across restarts.

# Stretch Cluster: 3 Availability Zones Split Brain Problem

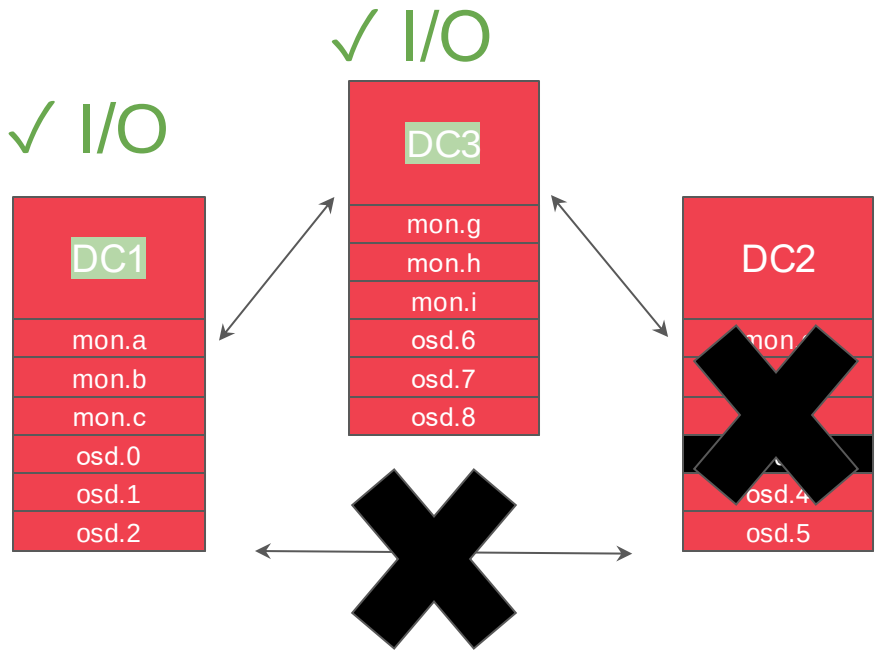



1. Netsplit detected between dc1 and dc2
2. Heuristic Calculation determines that we should choose DC1 + DC3 to be the surviving site since at has greater OSD count than D2 + DC3

DC1+DC3 OSD_COUNT = 6
DC2+DC3 OSD_COUNT = 5

DC1+DC3 should survive!

# Stretch Cluster: 3 Availability Zones Split Brain Problem




1. Netsplit detected between dc1 and dc2
2. Heuristic Calculation determines that we should choose DC1 + DC3 to be the surviving site since at has greater OSD count than D2 + DC2

DC2 is marked down and kept out of quorum, if the netsplit is no longer detected, then DC2 will stopped being marked down and rejoin the cluster!



# Thank you!

# Q & A

Links to 3AZ PRs:

- https://github.com/ceph/ceph/pull/56233
- https://github.com/ceph/ceph/pull/57381
- https://github.com/ceph/ceph/pull/60631
- https://github.com/ceph/ceph/pull/59248