BeeGFS

Storage System Availability

BeeGFS.io

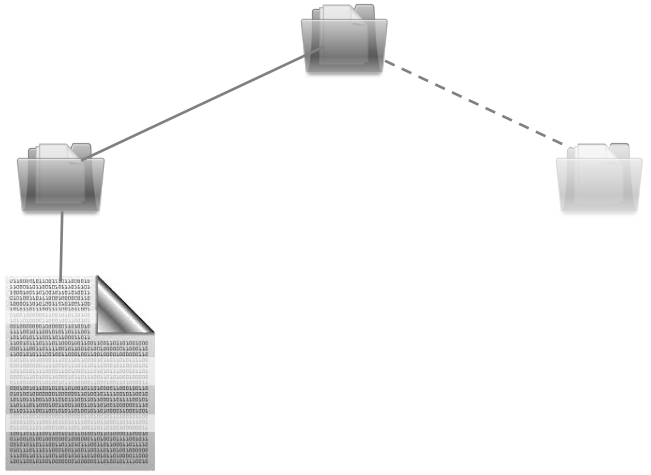
2022

Agenda

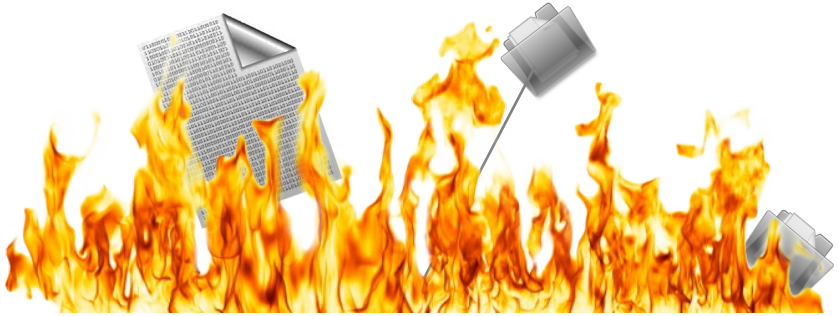
- Basic Concepts
  - Data availability
  - Data loss
- Buddy Mirror
  - Metadata
  - Storage

Data Availability vs. Data Loss

- Not available means…
  - I can‘t access (parts of) my data
  - I need to fix hardware or software
  - Jobs will hold until server with data is back up again

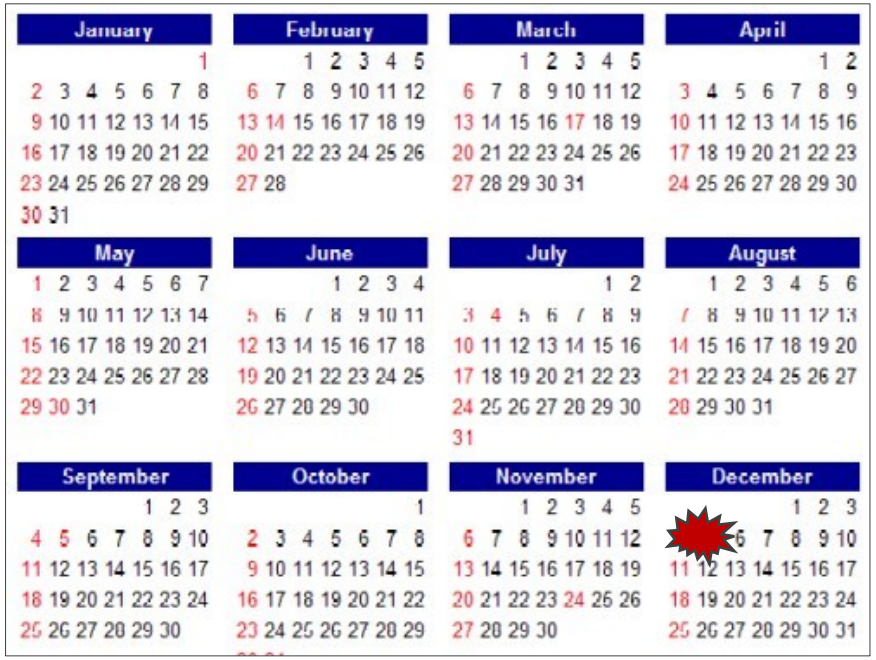


- Data loss means…
  - Data is damaged
  - I need to recover data from my backup
  - I have to recalculate results
  - Running beegfs-fsck might help
  - Damage might be irreparable



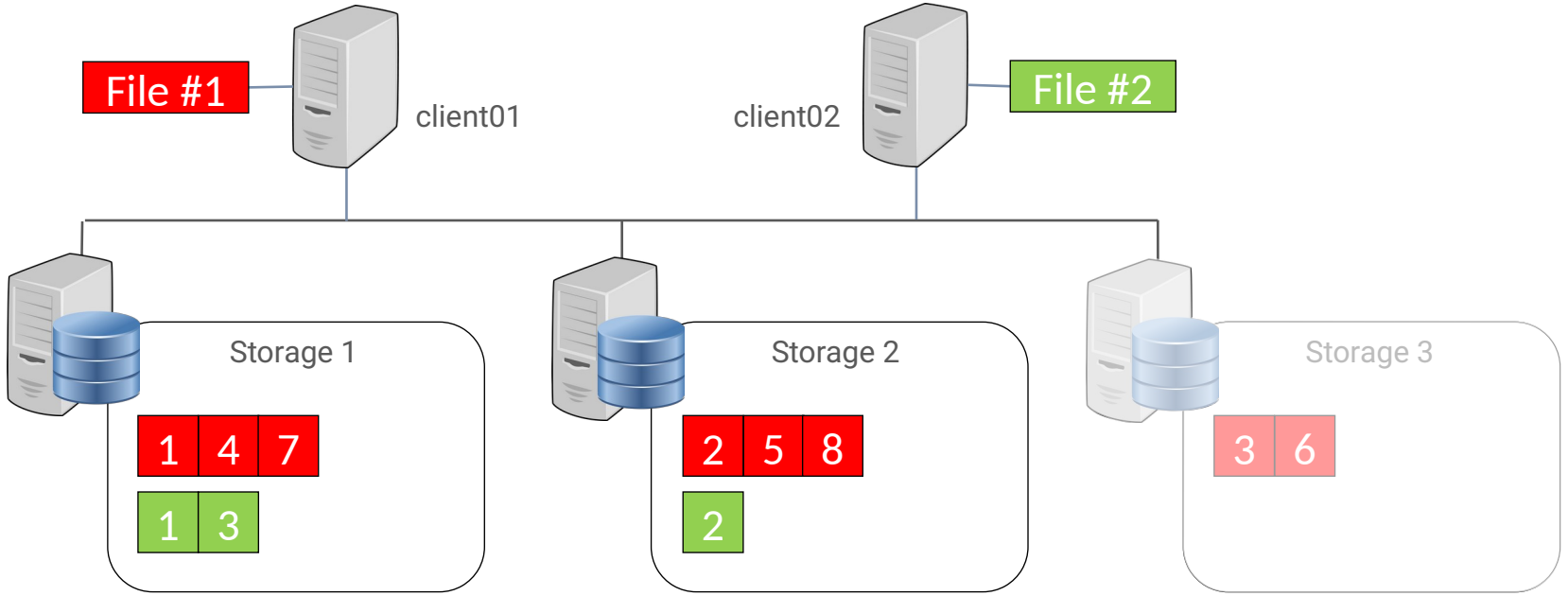
How do most BeeGFS users deal with availability?

- Most customers operate BeeGFS as their “work file system“
- Primary focus is performance for working data
- Why don‘t users care more about availability?
  - Risk and cost of unavailability hard to estimate.
  - Small cost over time
  - It is all about „price/performance“
  - Lost data can be recreated in most cases



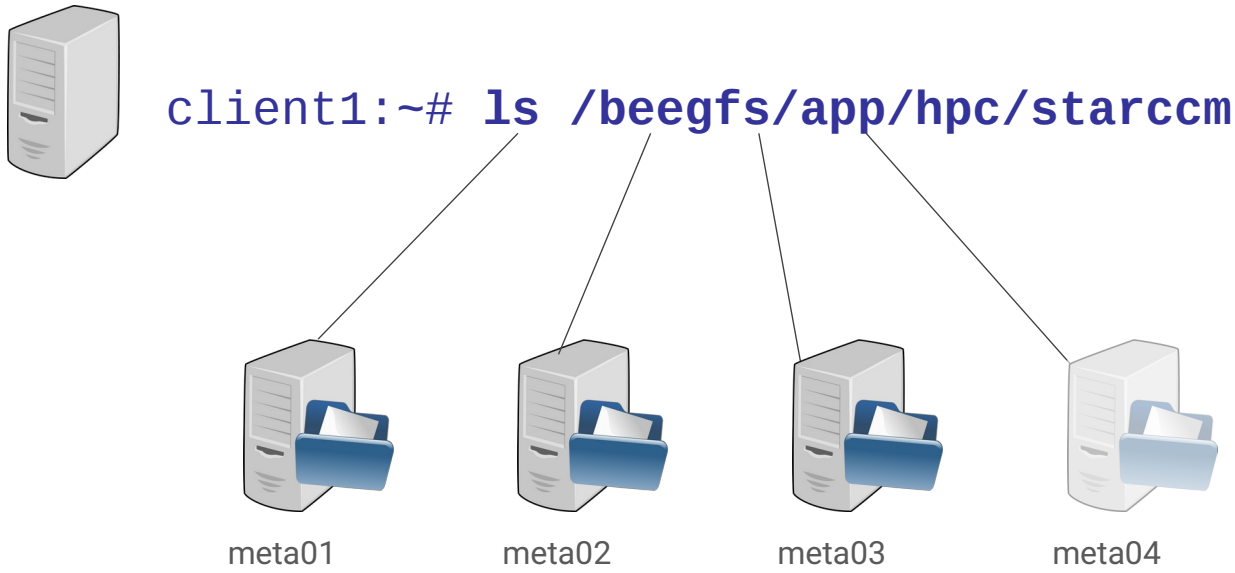
What happens if a BeeGFS server is down?

- Storage Server
  - You don‘t lose data
  - Clients which need data from the failed node wait until they run into timeout (configurable)
  - Clients which don‘t need data from the failed node will operate without problem



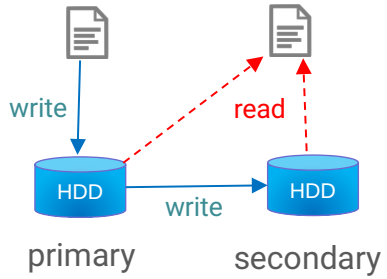
What happens if a BeeGFS server is down?

- Metadada Server
  - You don‘t lose data
  - Clients which don‘t need data from the failed node will operate without problem
  - Clients which need data from the failed node wait until they run into timeout (configurable)



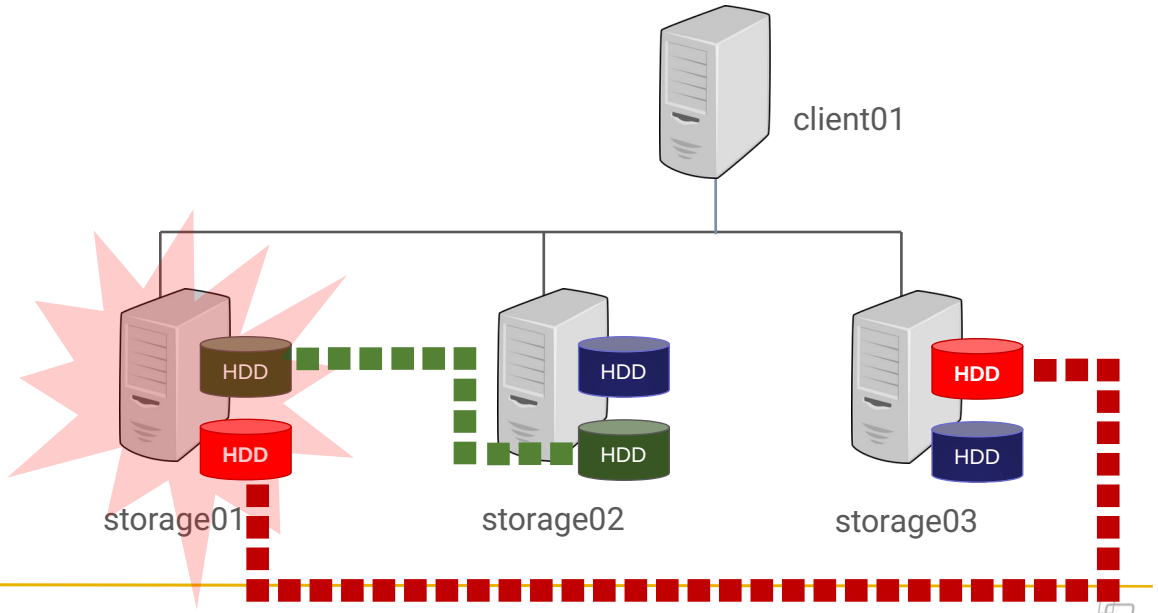
BeeGFS resiliency – Built-in Data Replication

- Buddy Group Mirroring
  - Data chunks mirrored among primary/secondary targets
  - Modifying operations sent to primary target and forwarded
  - Read possible from both targets



BeeGFS resiliency – Built-in Data Replication

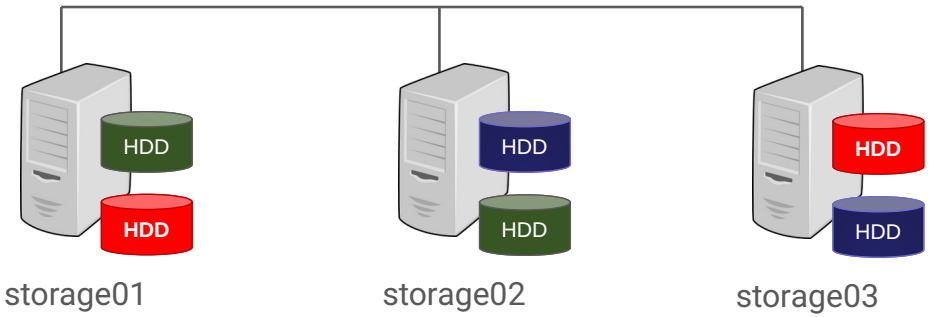
- Buddy Group Mirroring
  - Internal failover mechanism
    - In case primary is unreachable or fails, an automatic switch is performed
    - Self-healing (differential rebuild) when buddy comes back



BeeGFS resiliency – Built-in Data Replication

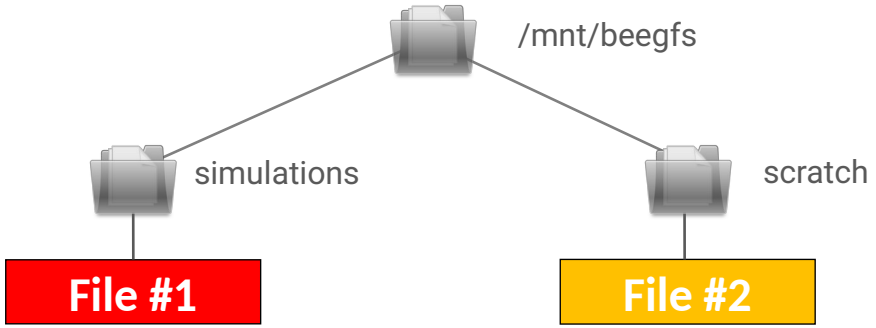
- Buddy Group Mirroring
  - First step: define buddy mirror groups
    - Option --automatic helpful for testing setups
    - Buddy targets should be in different servers

```
# beegfs-ctl --addmirrorgroup --nodetype=storage --automatic

# beegfs-ctl --addmirrorgroup --nodetype=storage --groupid=1
             --primary=101 --secondary=202

# beegfs-ctl --listmirrorgroups --nodetype=storage

  GroupID   PrimaryTarget  SecondaryTarget
  =======   =============  ===============
1 1         101            202
2 2         201            302
3 3         301            102
```

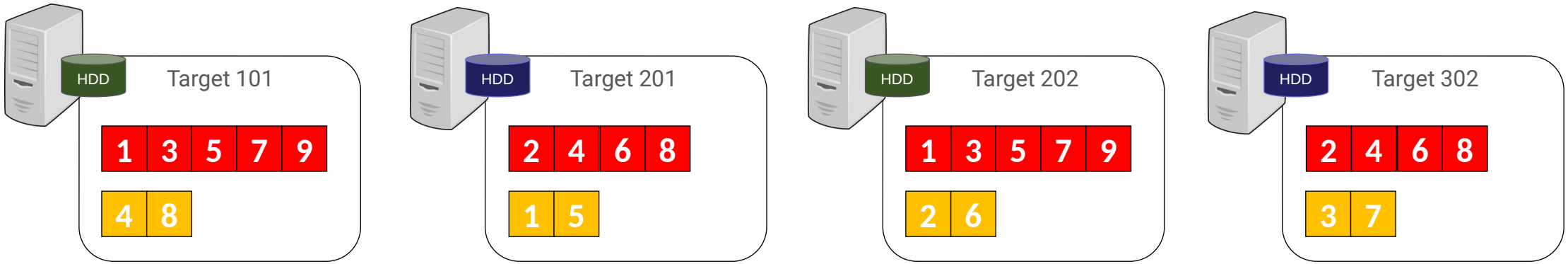


Buddy Mirroring Per Directory for storage

- **Buddy Group Mirroring**
  - Second step: set data striping pattern
  - Mirroring enabled globally or on a per-path basis
  - Targets can also store non-mirrored chunks




```
# beegfs-ctl --addstoragepool –desc=mirrorpool –mirrorgroups=1 --id=3

# beegfs-ctl --setpattern –storagepoolid=3  --pattern=buddymirror  --numtargets=1   /mnt/beegfs/simulations
```



Target States

- Two types of target state
  - Reachability state
  - Consistency state

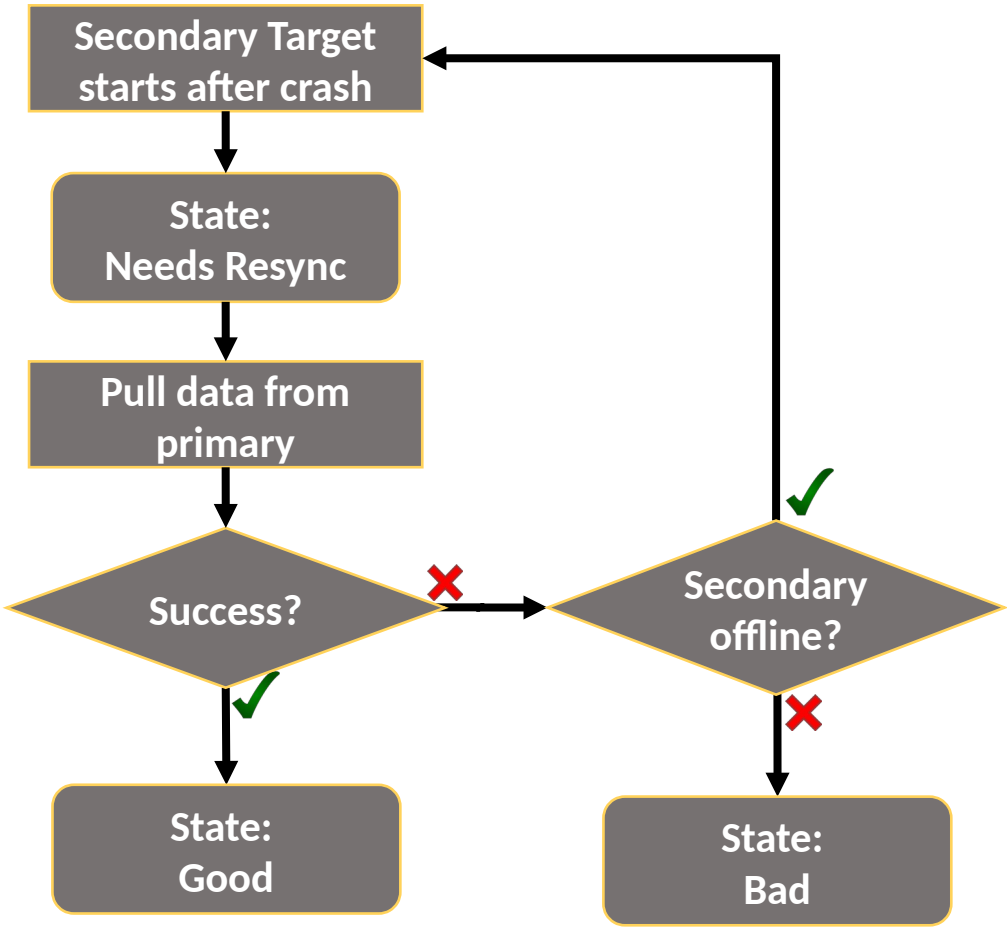
```
client01:~ # beegfs-ctl --listtargets  --state
             --nodetype=storage

TargetID       Reachability  Consistency  NodeID
========   ================  ===========  ======
101                  Online         Good       1
102                  Online         Good       1
201                  Online         Good       2
202                  Online         Good       2
301         Probably-Offline         Good       3
302                 Offline          Bad       3
401                  Online         Good       4
402                  Online         Good       4
403                  Online    Resyncing       4
403                  Online         Good       4
```

Reachability State

- Online
  - The target is reachable and fully usable by clients
- Probably Offline
  - The target might be offline
  - Intermediate state to avoid race conditions or split-brain
- Offline
  - If target is part of a buddy group, try a switchover

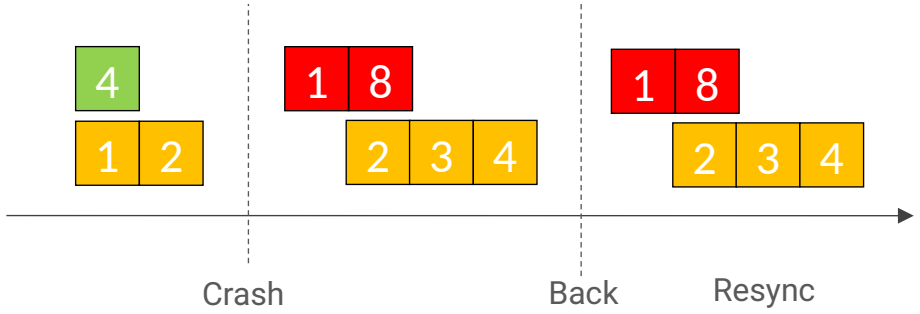
Consistency State

- Good
  - Target may be used without limitations
- Bad
  - A target is not reachable
  - A reync failed
  - Manual intervention might be needed
- Needs Resync
  - Secondary target needs a resync
  - Clients may still access non-mirrored files
- Resyncing
  - Resync in progress
  - Clients may still access non-mirrored files

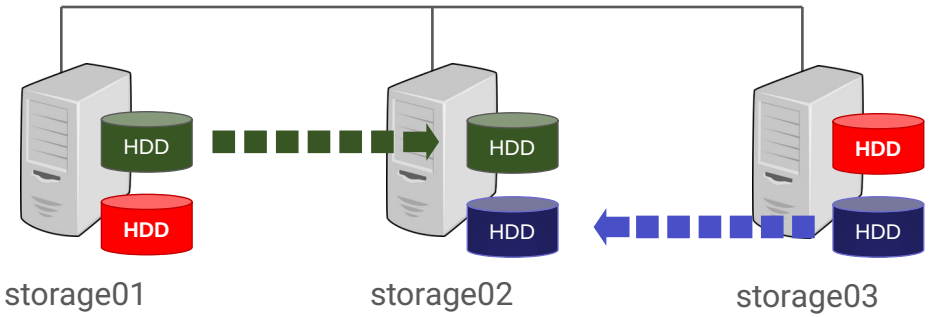


Resync

- Completely transparent & automatic when a node comes back after a failure
- Pushes all chunks modified since last successful communication from the primary the secondary
- Can also be triggered manually
  - Optionally: All files
    (e.g. rebuild of completely lost storage target)




```
# beegfs-ctl --startresync --mirrorgroup=2
             --timespan=5d
# beegfs-ctl --startresync --targetid=101
             --timespan=0d
```



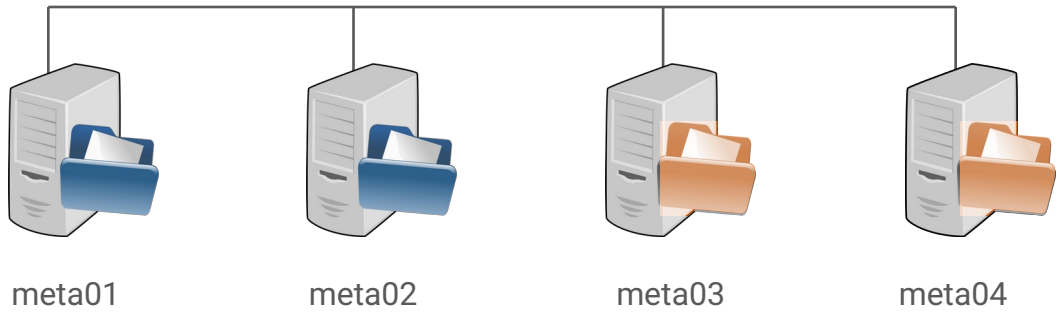
Metadata Mirroring

- Similar to storage data mirroring
  - Same basic commands

```
# beegfs-ctl --addmirrorgroup --nodetype=metadata --groupid=1 --primary=1 --secondary=2

# beegfs-ctl --listmirrorgroups --nodetype=metadata

GroupID   PrimaryTarget  SecondaryTarget
=======   =============  ===============
1         1              2
2         3              4
```

  - Same consistency and reachability states

```
client01:~ # beegfs-ctl --listtargets  --state --nodetype=metadata

TargetID       Reachability  Consistency  NodeID
========   ================  ===========  ======
1                    Online         Good       1
2                    Online    Resyncing       2
3                    Online         Good       3
4          Probably-Offline         Good       2
```



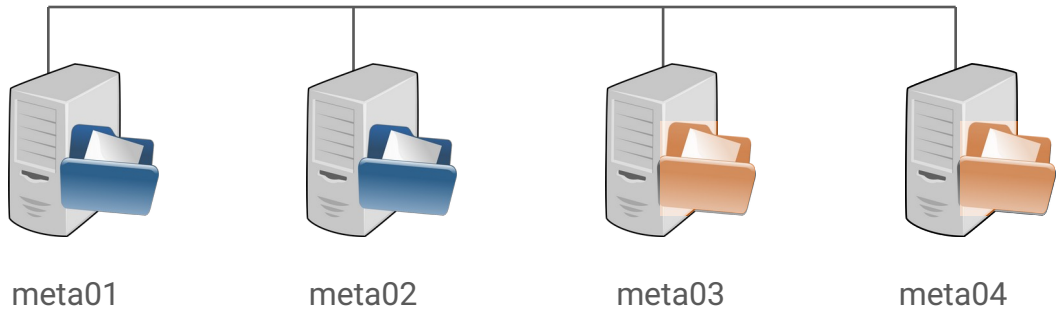
Metadata Mirroring

- Some differences
  - Nodes are buddies
  - Root node needs to be defined as primary
  - Activated by a special command

```
# beegfs-ctl --mirrormd
```

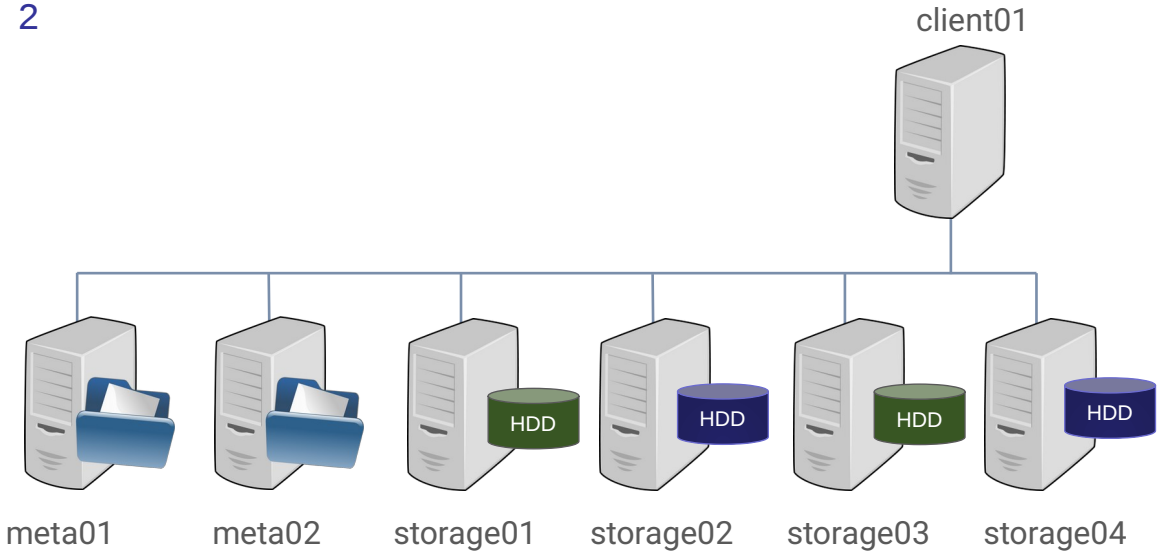
  - All clients must be stopped
  - Metadata must be restartet, then clients can be started again
  - Directories can have mirroring turned off

```
# beegfs-ctl --createdir --nomirror /beegfs/docs
```



Getting Information About Directory Entry

```
client01:~ # beegfs-ctl --getentryinfo --verbose /mnt/beegfs/simulations/file1.dat

Path: /simulations/file.dat
Mount: /mnt/beegfs
EntryID: 1-597EE887-1
Metadata buddy group: 100
Current primary metadata node: meta01 [ID: 1]
Stripe pattern details:
+ Type: Buddy Mirror
+ Chunksize: 1M
+ Number of storage targets: desired: 4; actual: 2
+ Storage mirror buddy groups:
  + 100
  + 200
Chunk path: u0/597E/E/0-597EE887-1/1-597EE887-1
Dentry path: 16/3F/0-597EE887-1/
```

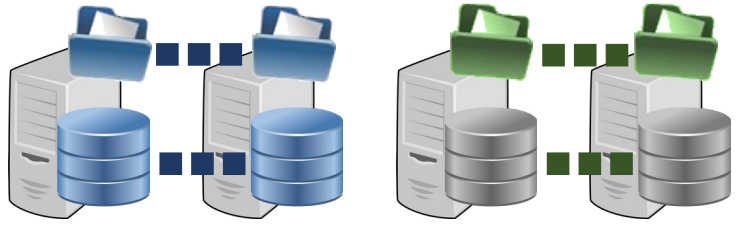


DIY

- **Setup BeeGFS buddymirror**
  - Buddy Group Mirroring
    - How many buddy groups should be defined?
    - Which services should be mirrored?
    - Which directories should be mirrored?

- **Define buddy groups in your test environment**
  - Both metadata and storage
  - Enable metadata mirroring
  - Enable storage mirroring for some directories

- **Test failover**



What's Next?

- Typical Administration Task ✓
- Sizing & Tuning ✓
- BeeGFS Resiliency Tools ✓
- BeeOND: BeeGFS on Demand
- BeeGFS Hive Index
- Conclusion

BeeGFS®

Thank You

Follow BeeGFS: