High Availability with BeeGFS

www.beegfs.com

Agenda

- Introduction
  - About Us
  - Key Aspects
  - Archictecture
- System Administration
  - Typical Administration Tasks
  - Advanced Configurations
  - Tools
- High Availability
  - Shared Storage
  - Buddy Mirroring
- Conclusion

Introduction

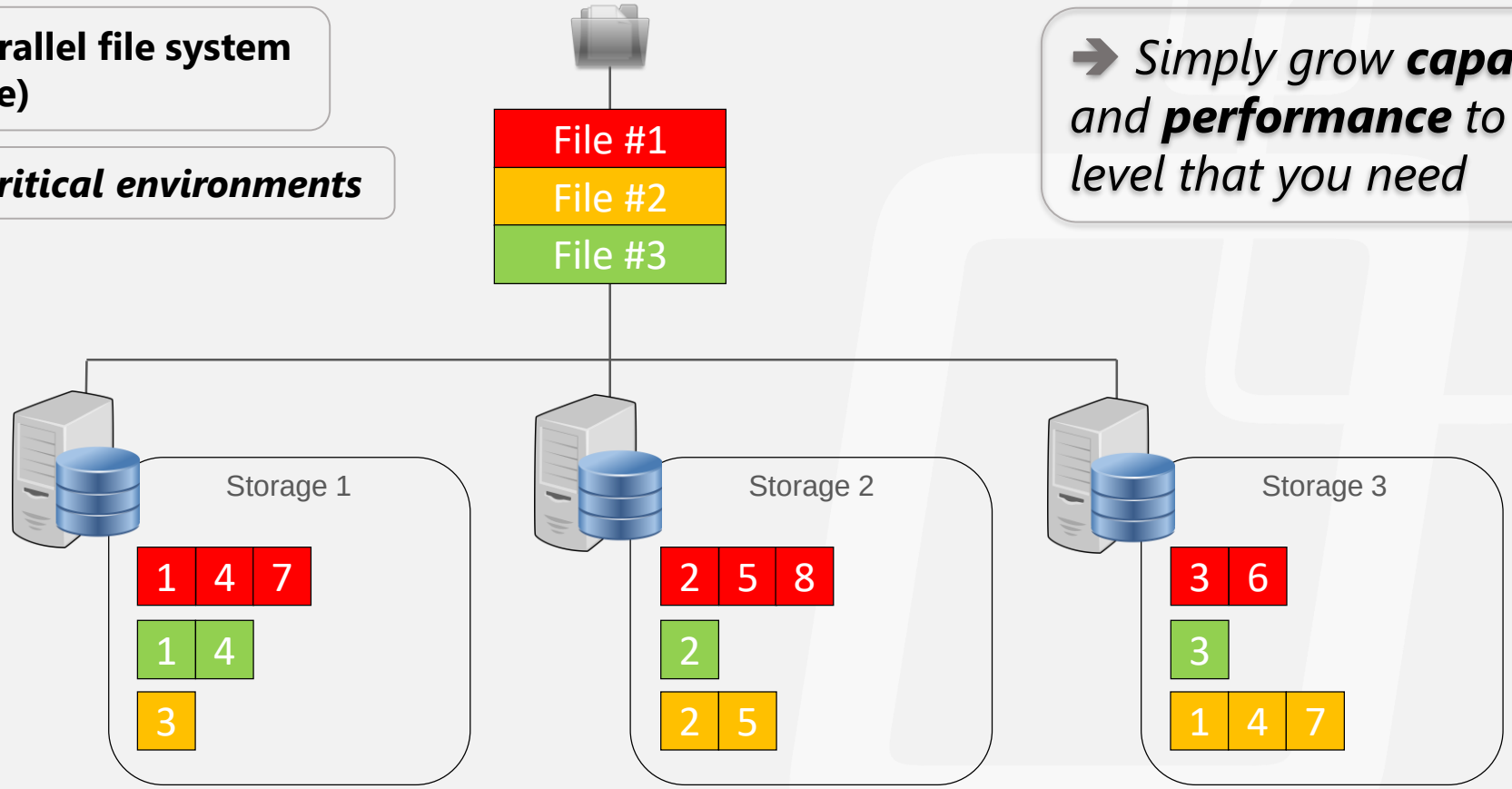
About Us
Key Aspects
Architecture

What is BeeGFS?

BeeGFS is…

…a *hardware-independent* parallel file system (aka Software-defined Storage)

…designed for *performance-critical environments*



➔ *Simply grow **capacity** and **performance** to the level that you need*

About Us

- BeeGFS was originally developed at the Fraunhofer Center for HPC

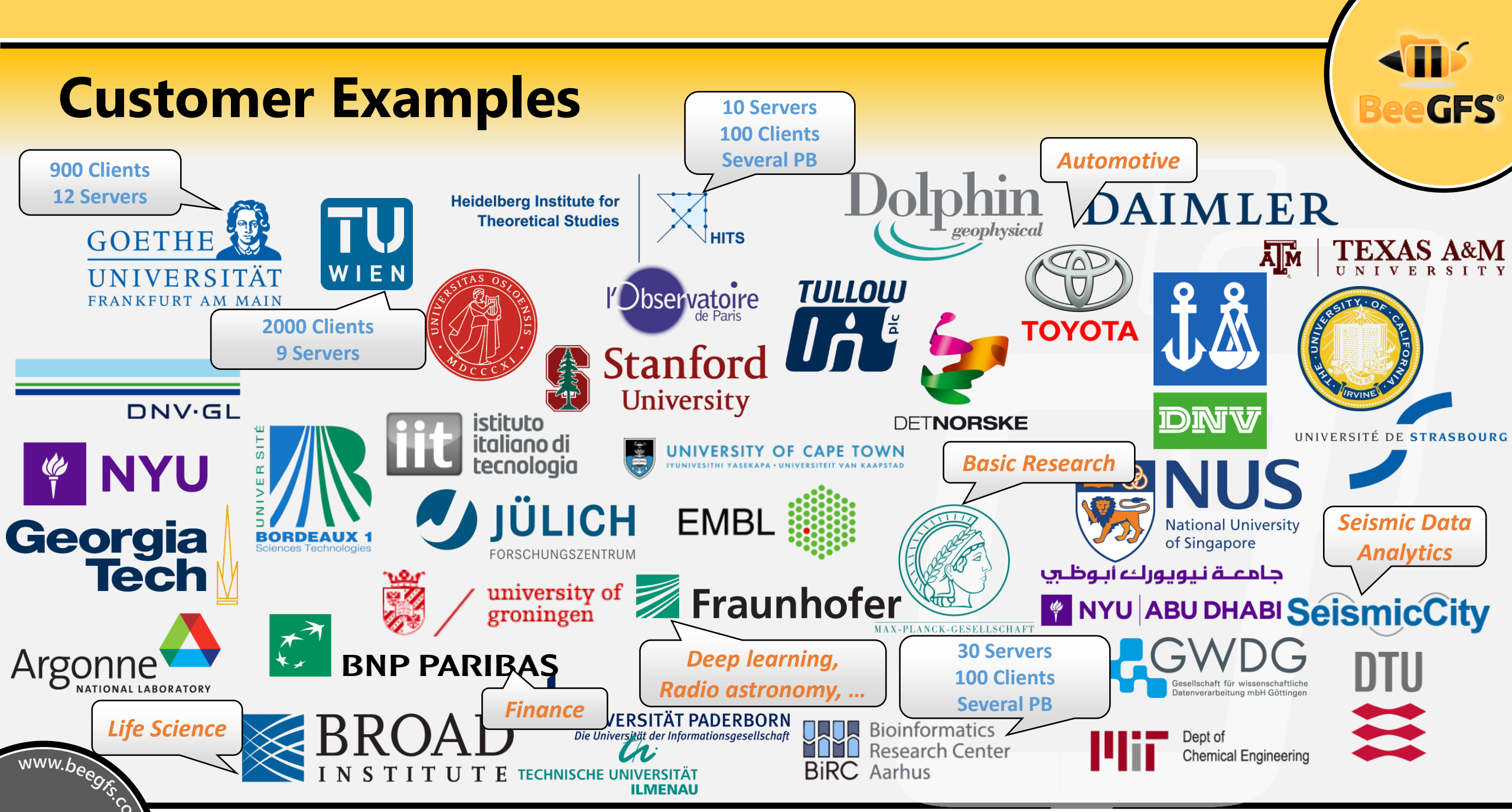
- The Fraunhofer Gesellschaft (FhG)
  - Largest organization for applied research in Europe
  - Special base funding by German government
  - Institutes, research units and offices around the globe
  - Staff: ~24000 employees

About Us

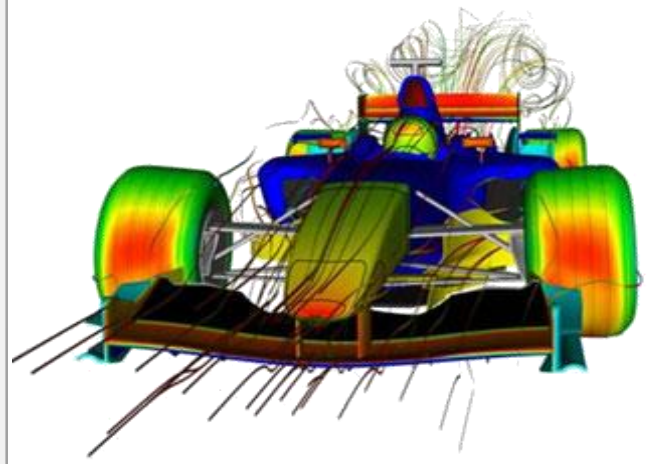
- ThinkParQ
  - A Fraunhofer spin-off
  - Founded in 2014 specifically for BeeGFS
  - Based in Kaiserslautern (right next to Fraunhofer HPC Center)
  - Consulting, professional services & support for BeeGFS
  - Cooperative development together with Fraunhofer
  - First point of contact for BeeGFS

Turn-key Sollution Patterns

Customer Examples



Key Aspects



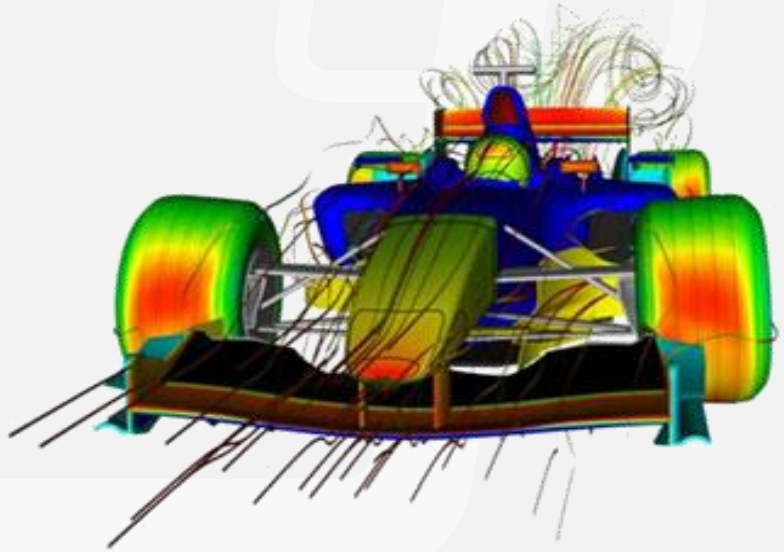
Maximum Performance & Scalability

High Flexibility

Robust & Easy to use

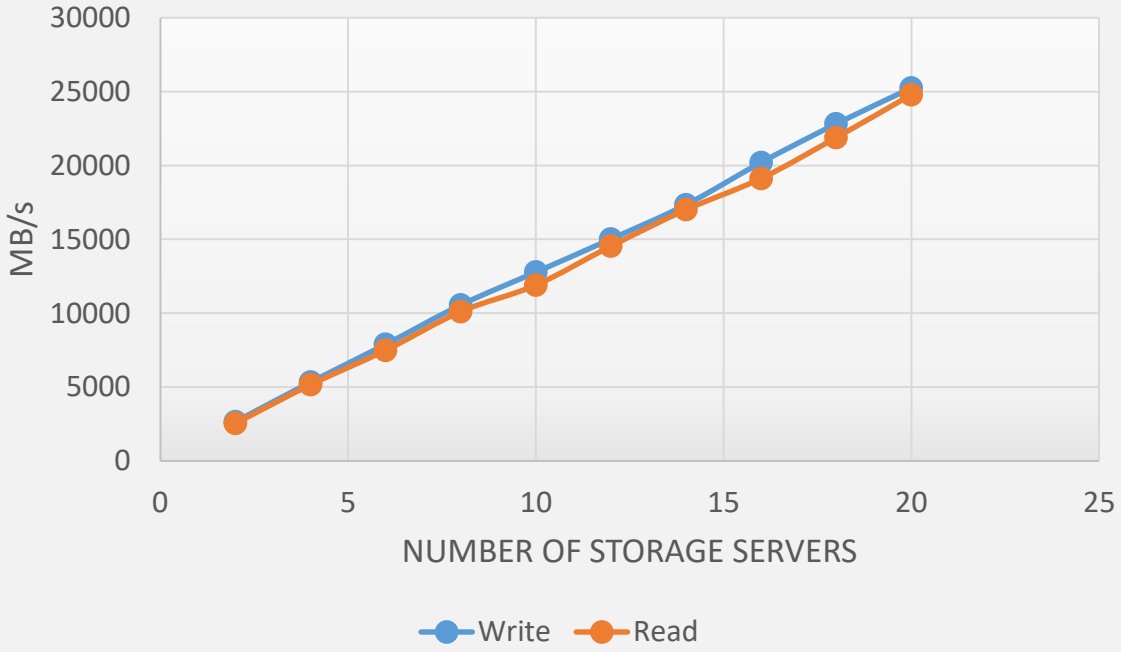
Key Aspects

- Performance & Scalability
  - Initially optimized for performance-critical workloads
  - Efficiently multi-threaded and light-weight design
    - "Not even breaking a sweat: BeeGFS at 10GB/s on single node all-flash unit over 100Gbit network" -ScalableInformatics
  - Supports RDMA/RoCE and TCP (Infiniband, Omni-Path, 100/40/10/1GbE, ...)
  - Distributed file contents & distributed metadata
  - Aggregated IOPS and throughput of multiple servers
  - Scales to millions of metadata operations per second
  - High single stream performance
    - 9GB/s single-stream throughput with Mellanox EDR (Few file streams completely saturate a 100GBit link.)

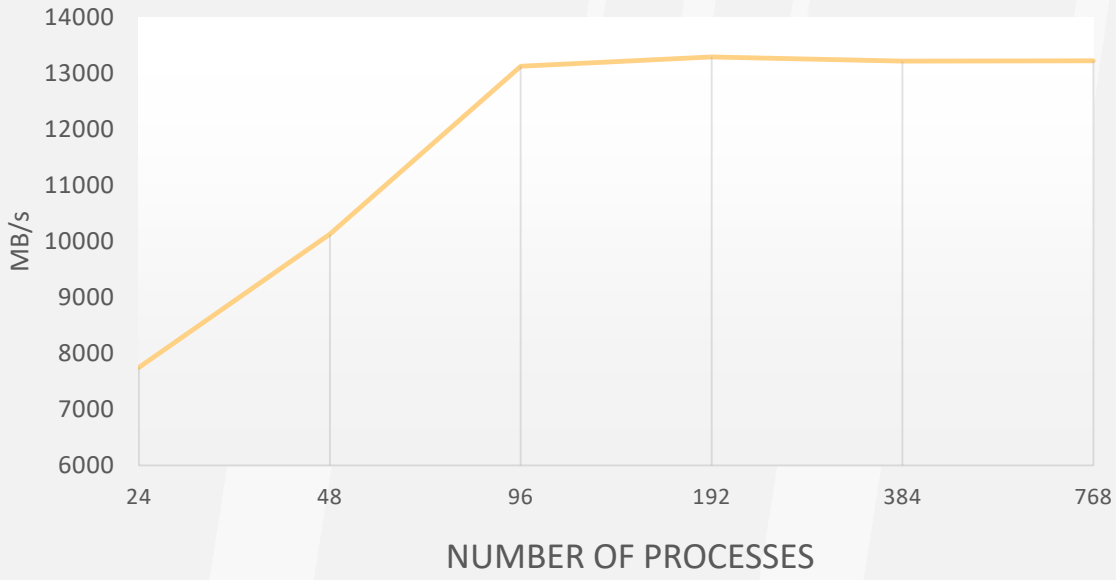


Throughput Scalability

Sequential read/write
up to 20 servers, 160 application processes



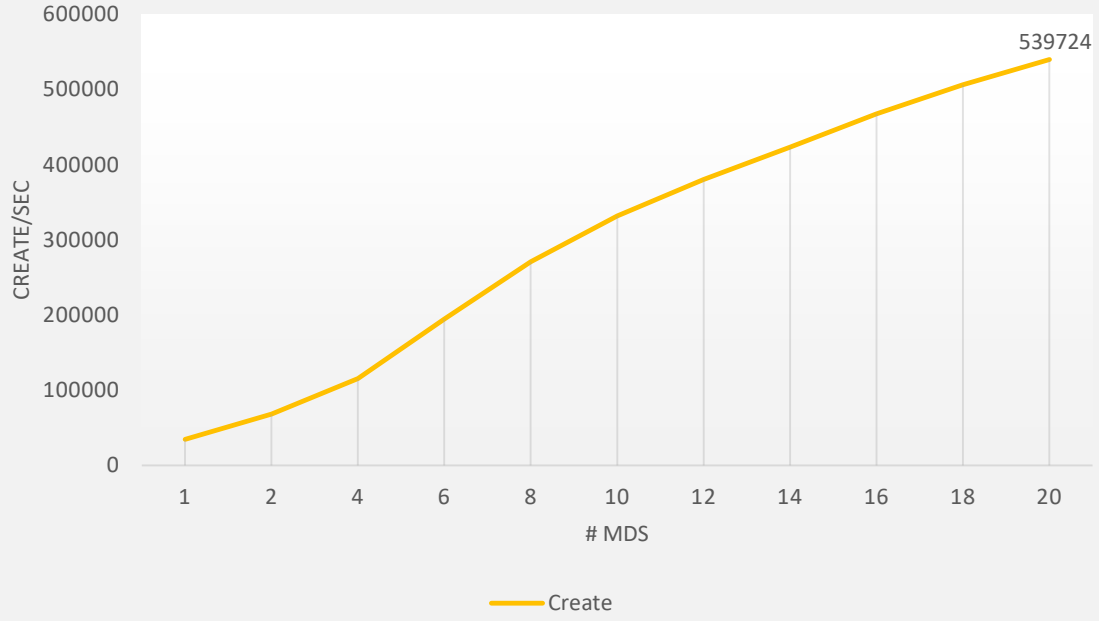
Strided unaligned shared file writes,
20 servers, up to 768 application processes



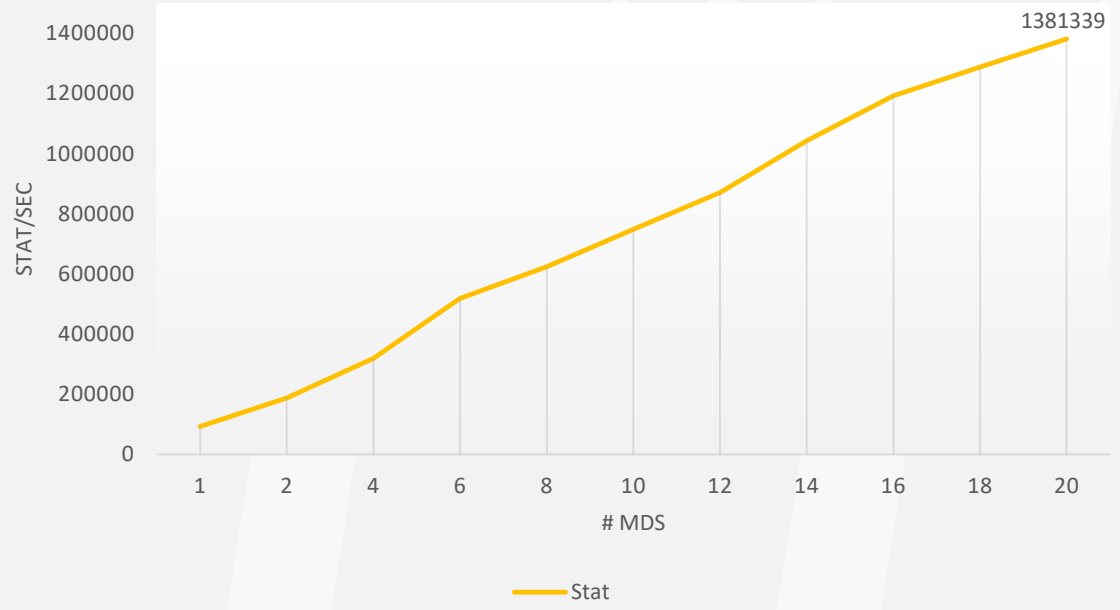
Note: Absolute numbers in these cases depend on per-server hardware performance, of course.

Metadata Scalability

File creation scalability with increasing number of metadata servers



File stat (attribute query) scalability with increasing number of metadata servers



Note: Absolute numbers in these cases depend on per-server hardware performance, of course.

Key Aspects

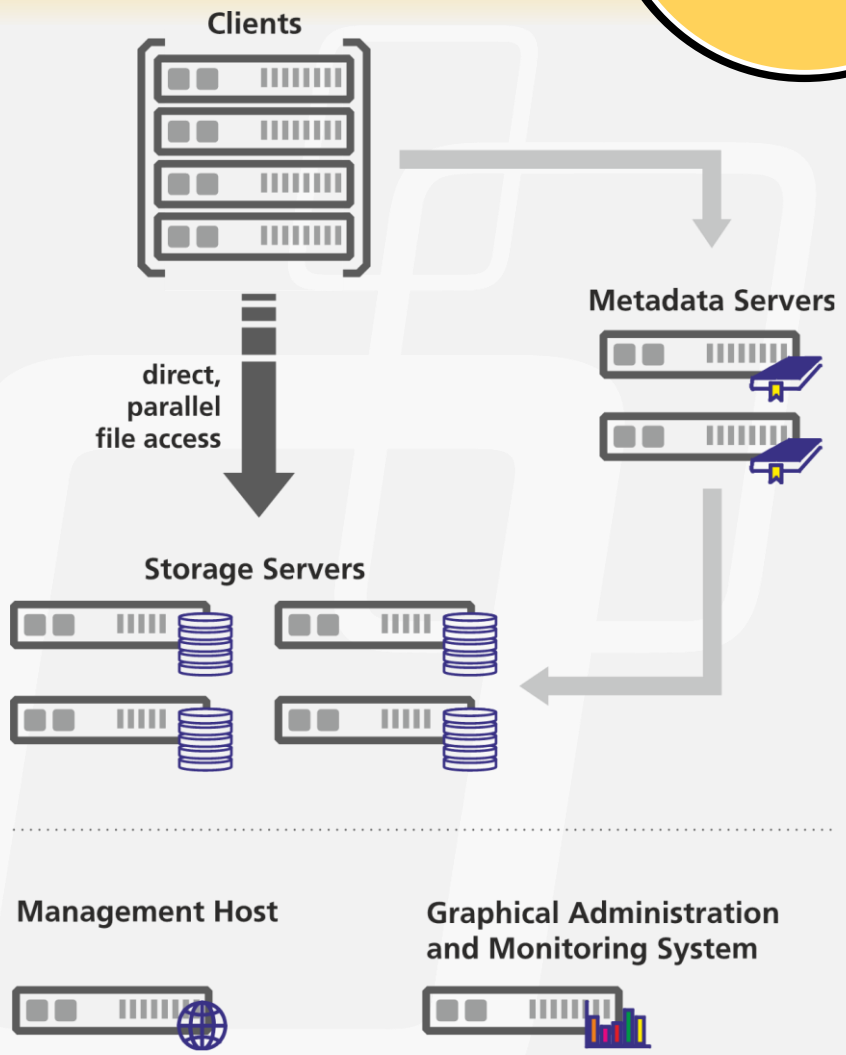
- Flexibility
  - Runs on different Architectures, e.g.:
  - No special hardware requirements
  - Packages for several Linux distributions and kernels:
  - Multiple BeeGFS services (any combination) can run together on the same machine
  - NFS & Samba re-export possible
  - Flexible data striping per-file / per-directory
  - Add servers or storage devices at runtime
  - Installation & updates without even rebooting

Key Aspects

- Robust & Easy to use
  - Very intensive suite of release stress tests, in-house production use before public release
    - The move from a 256 nodes system to a 1000 nodes system did not result in a single hitch, similar for the move to a 2000 nodes system.
  - Applications access BeeGFS as a normal (very fast) file system mountpoint
  - Servers run on top of standard local filesystems (ext4, xfs, zfs, ...)
  - No kernel patches
    - Updates of system packages, kernel and BeeGFS are trivially simple
  - Graphical tools
  - Comprehensive documentation (online, built-in)

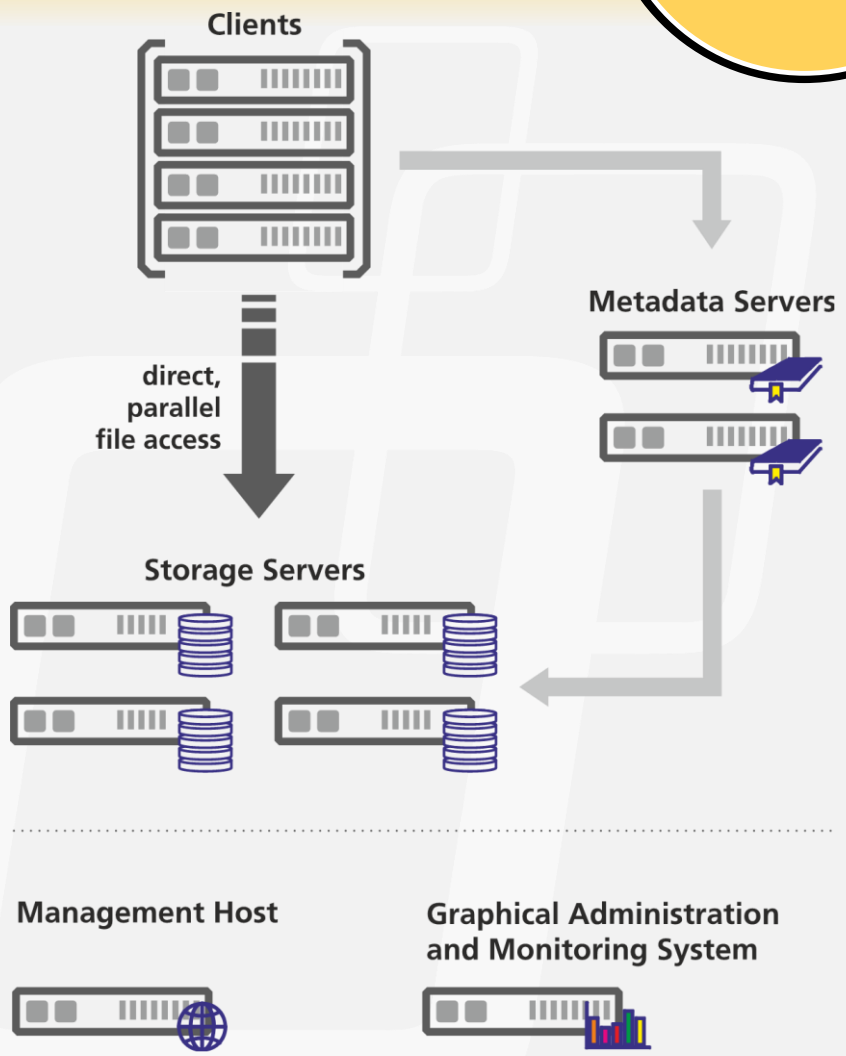
BeeGFS Architecture

- Management Service
  - Rendevouz point for (new) servers and (new) clients
  - Watches registered services and check their state
  - Not critical for performance, stores no user data
- Storage Service
  - Stores user file contents (data chunk files)
  - One or multiple storage services per BeeGFS instance
  - Manages one or more storage targets
    - In general, any directory on an existing local file system
    - Typically a RAID volume, either internal or externally attached



BeeGFS Architecture

- Metadata Service
  - Stores information about the data
    - Directory information
    - File and directory ownership
    - Location of user data files on storage targets
  - Not involved in data access between file open/close
- Client Service
  - Native Linux module to mount the file system
- Graphical Administration and Monitoring System
  - Administrative tasks and system information monitoring



System Administration

Typical Administration Tasks
Advanced Configurations
Tools

Installation

- Management service
  - No special requirements
  - Non performance-critical
  - Typically runs on a cluster master node or on a storage node

```
server01:~ # yum install beegfs-mgmtd

server01:~ # beegfs-setup-mgmtd -p /data/beegfs/beegfs_mgmtd

server01:~ # systemctl start beegfs-mgmtd
```

server01

Installation

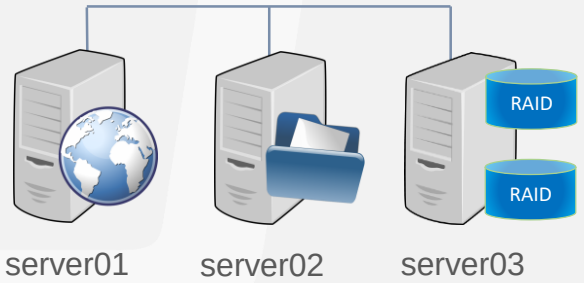
- Metadata Service
  - Storage device: SSDs widely used, RAID1 for low latency
  - Local file system: ext4 (fast handling of small files)
  - Capacity: 0.5% of total storage space
  - Processor clock frequency more important than number of CPU cores for low latency
  - Number of CPU cores important for systems with lots of clients
  - Low latency network important for metadata access
  - RAM used for caching – The more the better

```
server01:~ # yum install beegfs-meta

server01:~ # beegfs-setup-meta -p /mnt/ssd1 -s 1 -m server01

server01:~ # systemctl start beegfs-meta
```

Installation

- Storage Service
  - Multiple storage targets: RAID6 volumes, 12 disks per volume
  - Local file systems: usually XFS, but also others like ZFS and ext4
  - Mid-range CPUs are sufficient
  - Low latency high throughput network recommended
  - RAM used for caching – The more the better

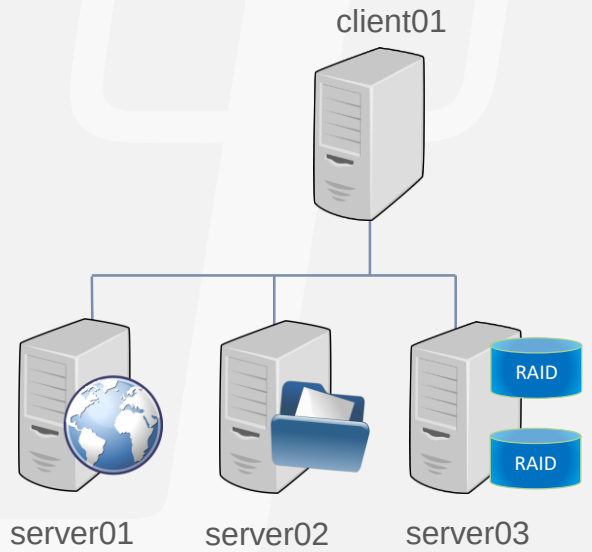
```
server03:~ # yum install beegfs-storage
server03:~ # beegfs-setup-storage -p /mnt/raid1 -s 3 -i 301 -m server01
server03:~ # beegfs-setup-storage -p /mnt/raid2 -s 3 -i 302 -m server01
server03:~ # systemctl start beegfs-storage
```



Installation

- Client Service
  - The only kernel module
  - Automatically build for newly installed kernels
  - Low latency high throughput network recommended

```
client01:~ # yum install beegfs-client

client01:~ # beegfs-setup-client -m server01

client01:~ # systemctl start beegfs-client
```



Updating, upgrading, and versioning

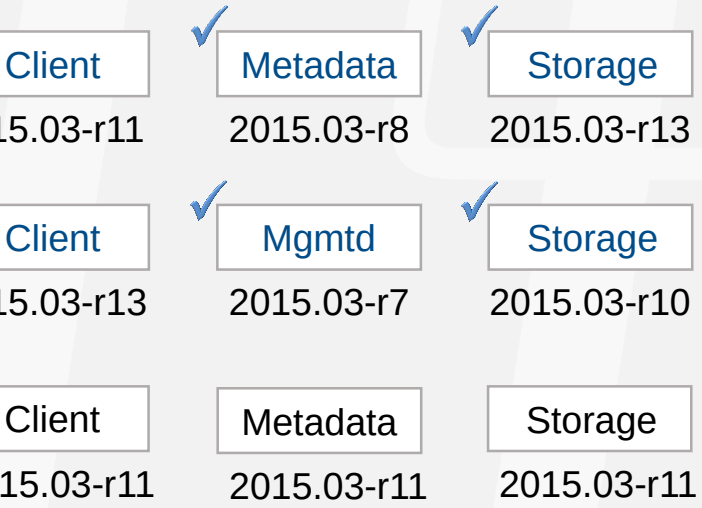
- Version scheme

Previous			Current		
2015.03	-	r20	6	.	4
major		minor	major		minor

- Minor release update
  - Minor releases can be mixed
  - Downtime unnecessary
  - Service restart required

```
# yum update beegfs-mgmtd

# systemctl restart beegfs-mgmtd
```

- Major release upgrade
  - Major releases cannot be mixed
  - Downtime needed



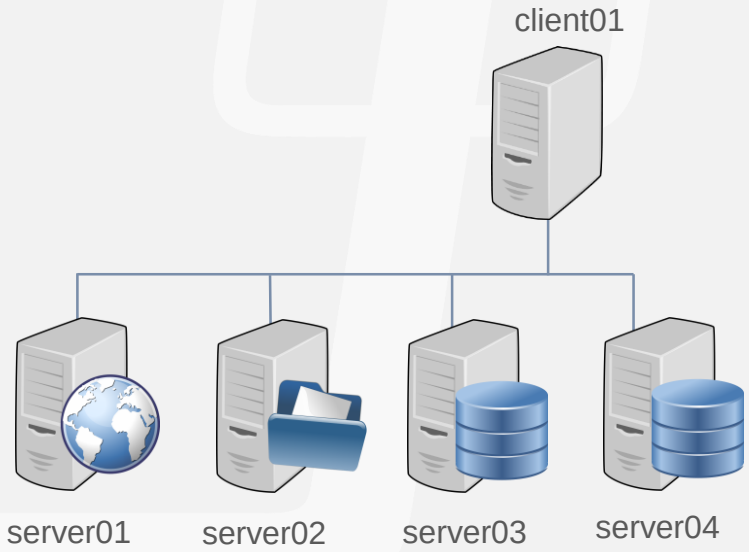
Command beegfs-ctl

```
client01:~ # beegfs-ctl --help
BeeGFS Command-Line Control Tool (http://www.beegfs.com)

GENERAL USAGE:
  $ beegfs-ctl --<modename> --help
  $ beegfs-ctl --<modename> [mode_arguments] [client_arguments]

MODES:
  --listnodes           => List registered clients and servers.
  --listtargets         => List metadata and storage targets.
  --removenode (*)      => Remove (unregister) a node.
  --removetarget (*)    => Remove (unregister) a storage target.
  --getentryinfo        => Show file system entry details.
  --find                => Find files located on certain servers.
  --migrate             => Migrate files to other storage servers.
  --serverstats         => Show server IO statistics.
  --clientstats         => Show client IO statistics.
  --userstats           => Show user IO statistics.
  --storagebench (*)    => Run a storage targets benchmark.
  --getquota            => Show quota information for users or groups.
  --setquota (*)        => Sets the quota limits for users or groups.
  --listmirrorgroups    => List mirror buddy groups.
  --addmirrorgroup (*) => Add a mirror buddy group.
  ...
```

Part of package beegfs-utils

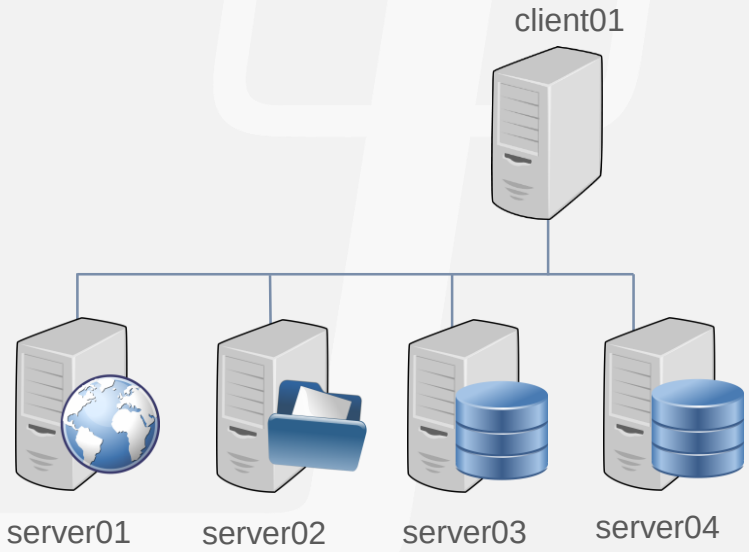
Listing registered nodes

```
client01:~ # beegfs-ctl --listnodes --nodetype=storage

server03 [ID: 3]
server04 [ID: 4]

client01:~ # beegfs-ctl --listnodes --nodetype=storage --nicdetails

server03 [ID: 3]
   Ports: UDP: 8003; TCP: 8003
   Interfaces:
   + ib0[ip addr: 10.12.20.3; type: RDMA]
   + ib0[ip addr: 10.12.20.3; type: TCP]
   + eth0[ip addr: 10.10.20.3; type: TCP]

server04 [ID: 4]
   Ports: UDP: 8003; TCP: 8003
   Interfaces:
   + ib0[ip addr: 10.12.20.4; type: RDMA]
   + ib0[ip addr: 10.12.20.4; type: TCP]
   + eth1[ip addr: 10.10.20.4; type: TCP]
```



Checking servers

```
client01:~ # beegfs-check-servers

Management
==========
server01 [ID: 1]: reachable at 10.12.20.3:8008 (protocol: TCP)

Metadata
==========
server02 [ID: 2]: reachable at 10.12.20.4:8005 (protocol: RDMA)

Storage
==========
server03 [ID: 3]: reachable at 10.12.20.3:8003 (protocol: RDMA)
server04 [ID: 4]: reachable at 10.12.20.4:8003 (protocol: RDMA)
```

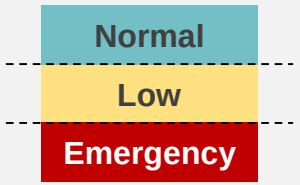


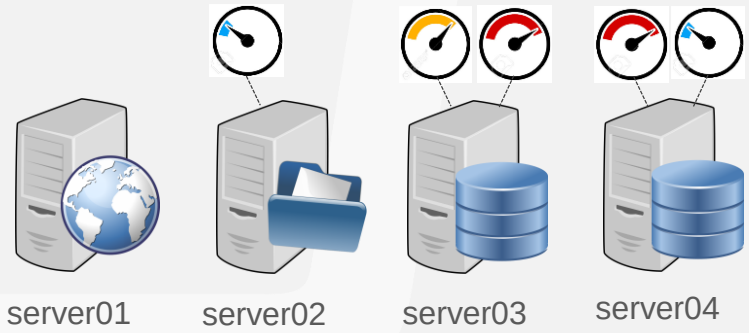
Checking free space

```
client01:~ # beegfs-df

METADATA SERVERS:
TargetID        Pool        Total         Free      InodesTotal      InodesFree
========        ====        =====         ====      ===========      ==========
       2    [normal]     240.0GB      230.0GB           158.8M           148.7M

STORAGE TARGETS:
TargetID         Pool        Total         Free      InodesTotal      InodesFree
========  ===========        =====         ====      ===========      ==========
     301  [emergency]     9168.7GB       4.5GB           582.2M           87.2M
     302        [low]     9168.7GB     102.5GB           582.2M           42.0M
     401  [emergency]     9168.7GB       2.5GB           582.2M           75.2M
     402     [normal]     9168.7GB    2112.5GB           582.2M           92.2M
```

tuneStorageSpaceLowLimit = 200G
tuneStorageSpaceEmergencyLimit = 5G

Pools

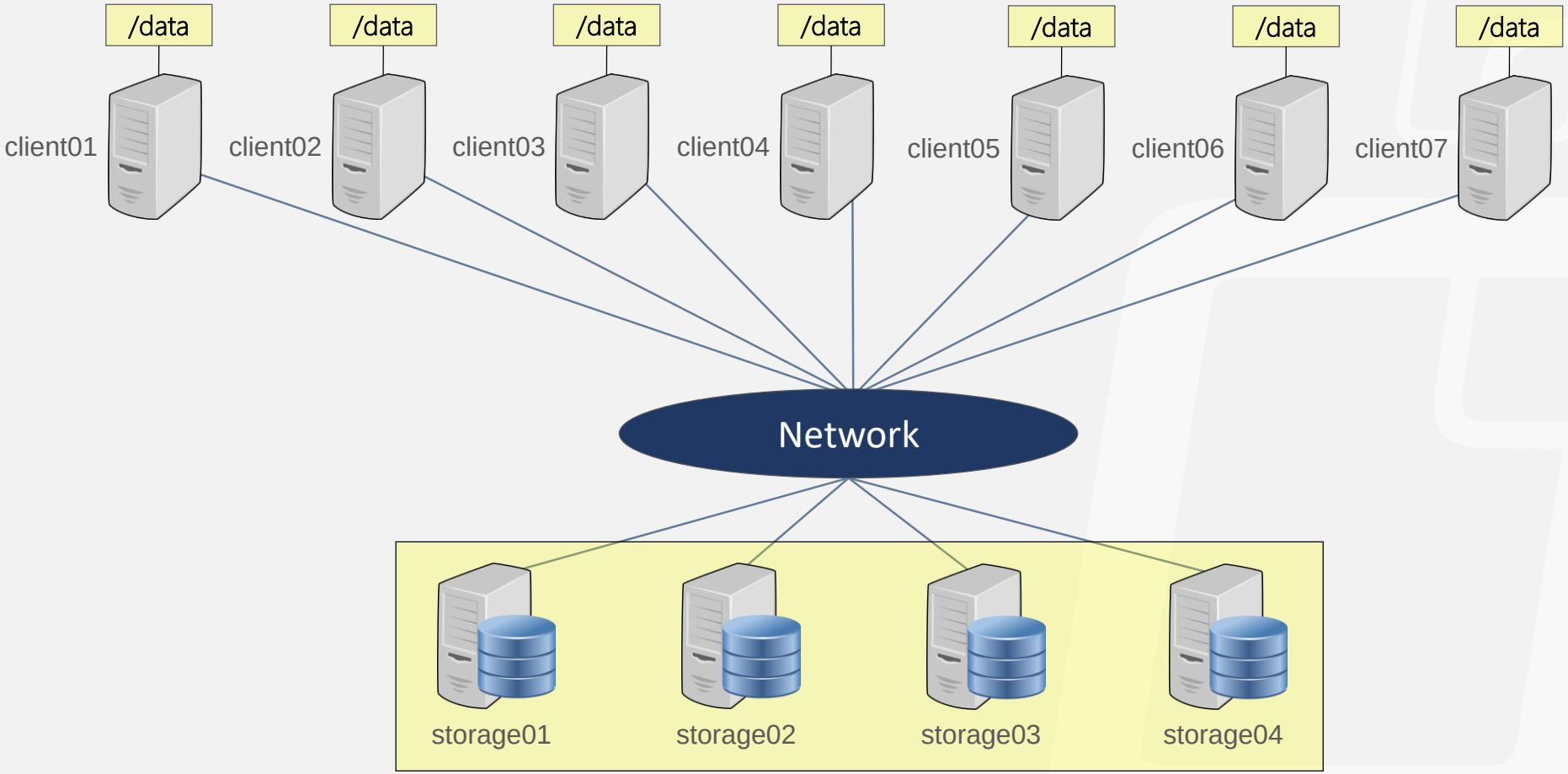




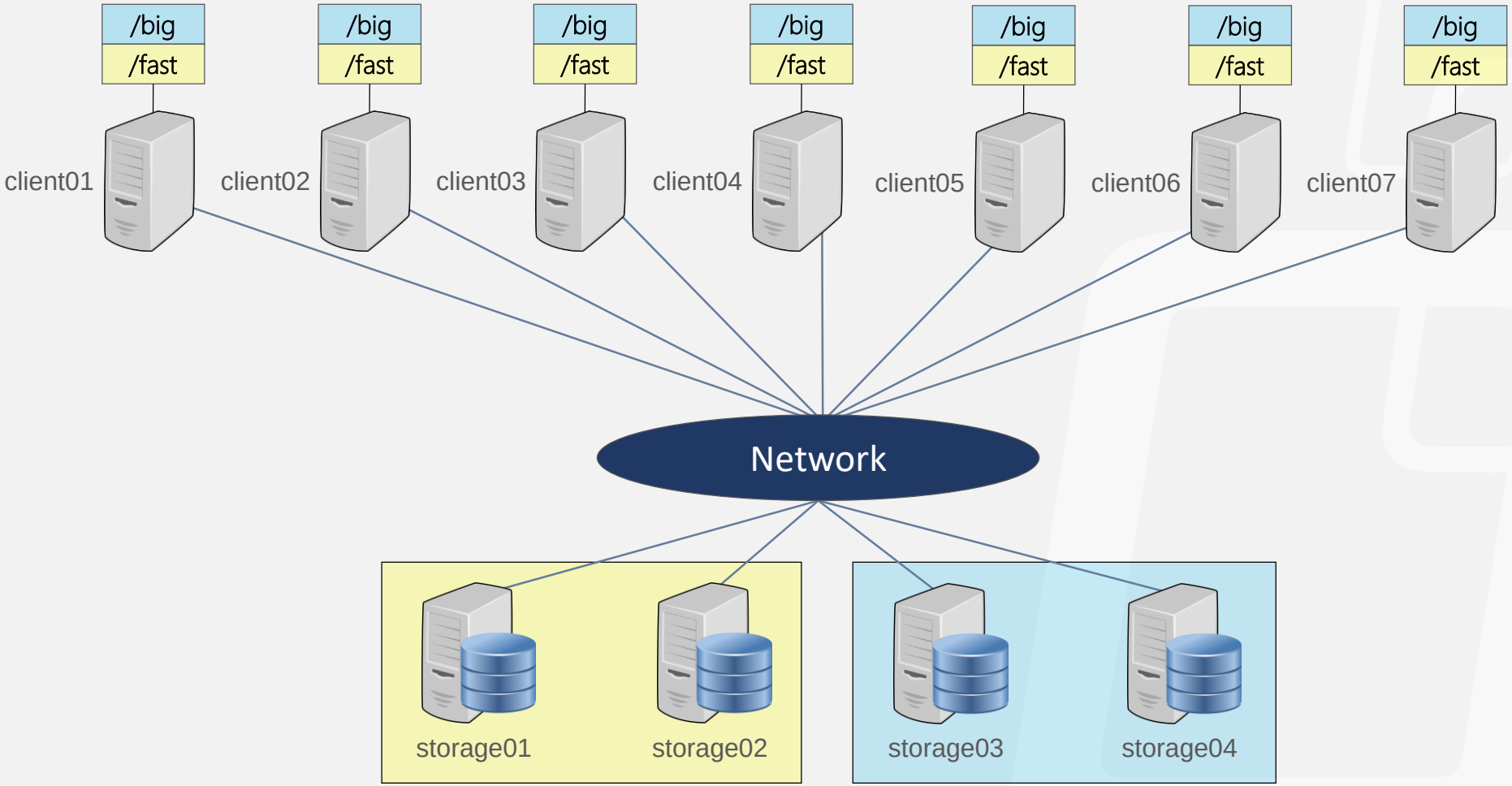
File system consistency check and repair

```
client01:~ # beegfs-fsck --checkfs [--readonly | --automatic]

--------------------------------------------------------------------
Started BeeGFS fsck in forward check mode [Mon Sep 29 21:47:57 2015]
Log will be written to /var/log/beegfs-fsck.log
Database will be saved as /var/lib/beegfs/beegfs-fsck.db
--------------------------------------------------------------------

Step 1: Check reachability of nodes: Finished
Step 2: Gather data from nodes:

Fetched data > Directory entries: 504598 | Inodes: 504598 | Chunks: 904831

Step 3: Check for errors...

* Target is used, but does not exist... Finished
* File has a missing target in stripe pattern... Finished
* Dentry-by-ID file is present, but no corresponding dentry... Finished
* Dentry-by-ID file is broken or missing... Finished
* Chunk is saved in wrong path... Finished
* Wrong owner node saved in inode... Finished
* Dentry points to inode on wrong node... Finished
...
```

Mount multiple BeeGFS instances



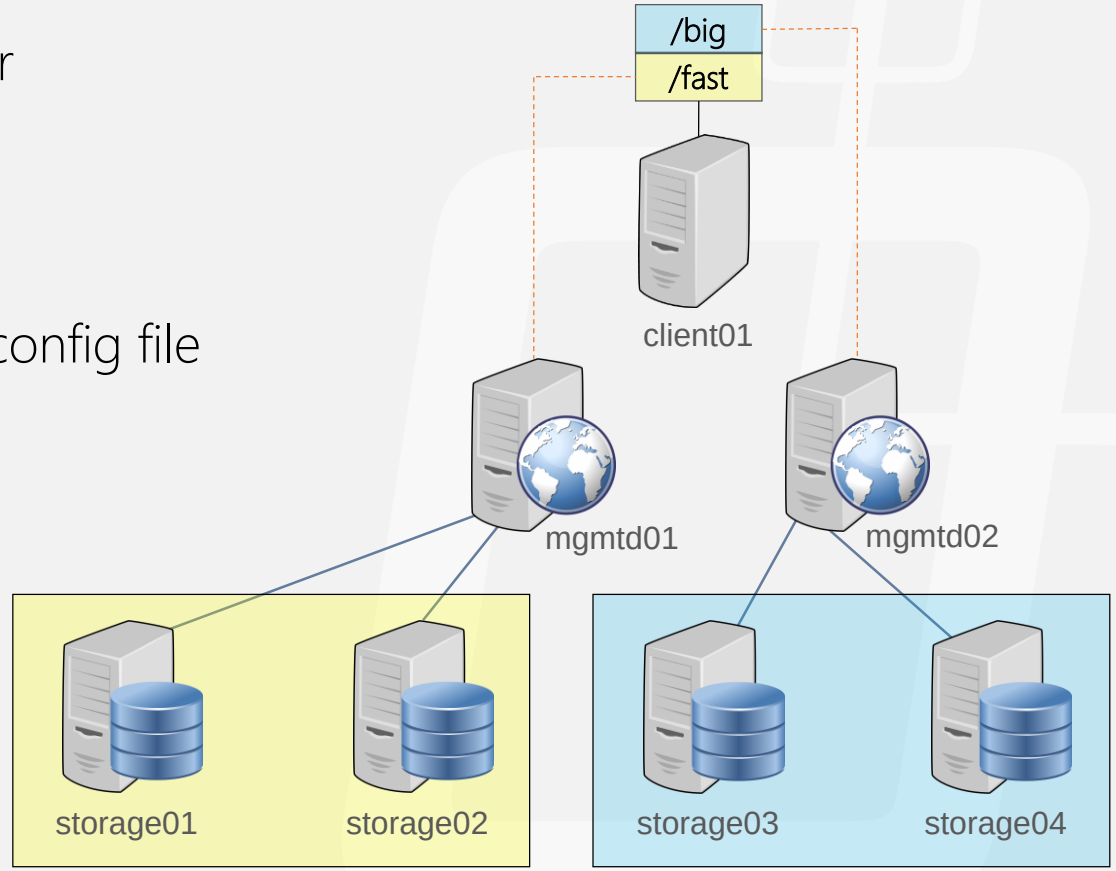
Mount multiple BeeGFS instances



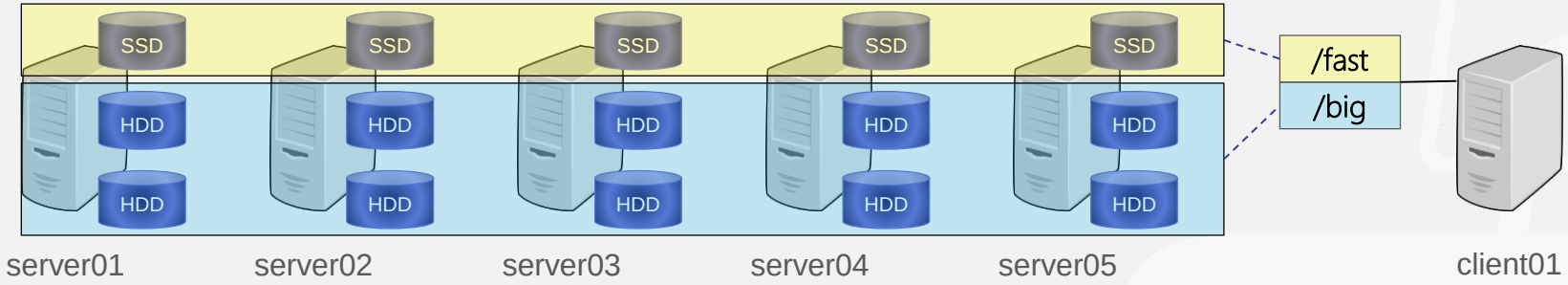
Mount multiple BeeGFS instances

- Create copies of /etc/beegfs/beegfs-client.conf
  - … with new names
  - … pointing to a second management server
- Edit /etc/beegfs/beegfs-mounts.conf
  - Map each mountpoint to a different client config file

```
/fast /etc/beegfs/beegfs-client-fast.conf
/big  /etc/beegfs/beegfs-client-big.conf
```



Multiple BeeGFS instances on the same servers



- Enable “multi-mode” in /etc/default/beegfs-...

```
server01:~ # cat /etc/default/beegfs-storage
MULTI_MODE="YES"
```

- Create a separate directory in /etc/beegfs for each instance with config files

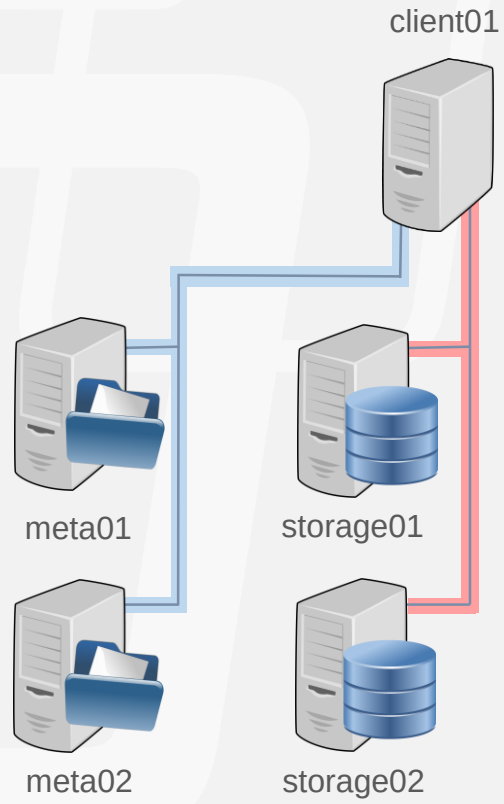
```
/etc/beegfs/fast.d
/etc/beegfs/big.d
```

More convenient in future versions

Limit traffic to specific network interfaces

- Use options in files /beegfs/beegfs-*.conf
  - connInterfacesFile: Registered interfaces for incoming connections
  - connNetFilterFile: Limit outgoing connections to certain IP address ranges

```
client01:~ # beegfs-ctl --listnodes --nodetype=client --nicdetails

14E8-53E4DE10-demo-io1
   Ports: UDP: 18004; TCP: 0
   Interfaces:
   + eth0[ip addr: 10.10.201.203; type: TCP]

FC13-53EB9471-demo-io2
   Ports: UDP: 18004; TCP: 0
   Interfaces:
   + ib0[ip addr: 10.12.201.204; type: RDMA]
   + ib0[ip addr: 10.12.201.204; type: TCP]
   + eth1[ip addr: 10.10.201.204; type: TCP]

Number of nodes: 2
```

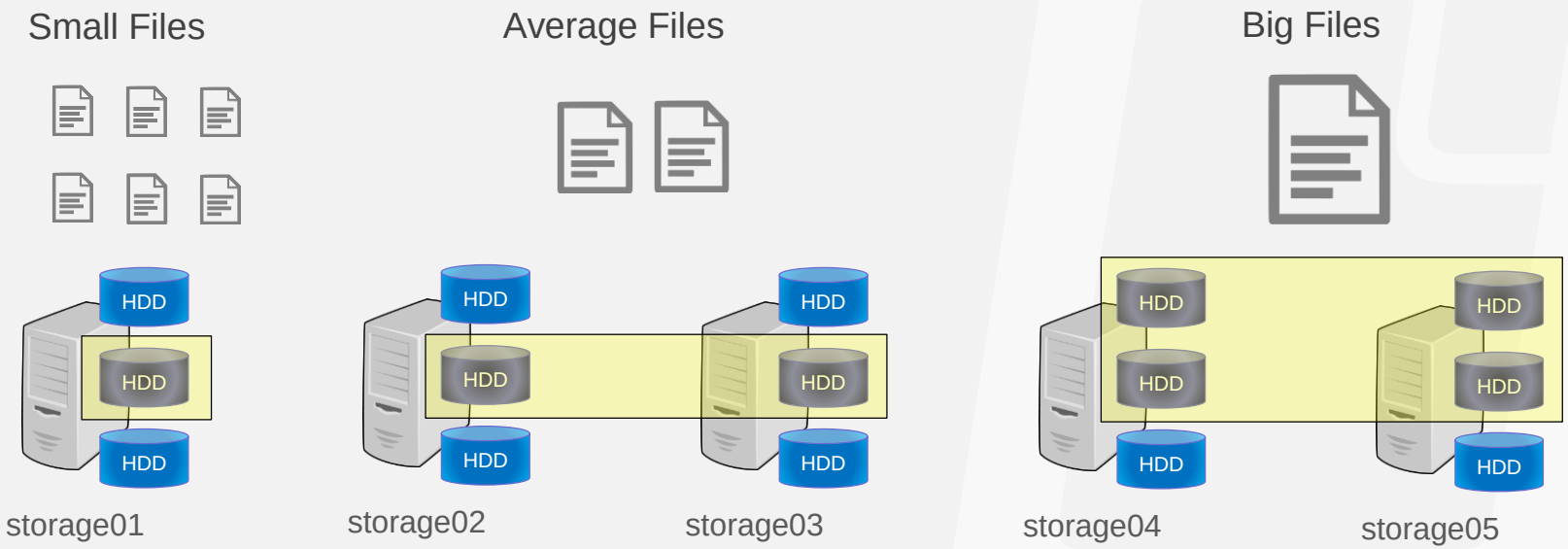


Data Striping

- Setting Data Striping Pattern per Path

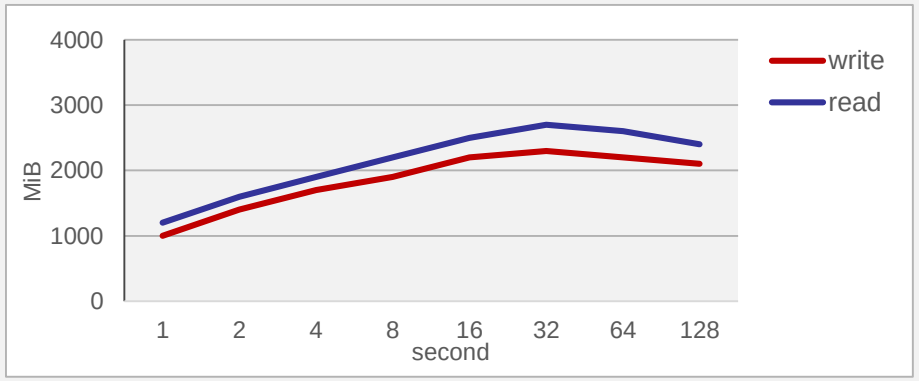
```
client01:~ # beegfs-ctl --setpattern --chunksize=1m --numtargets=4 /data/simulations
```

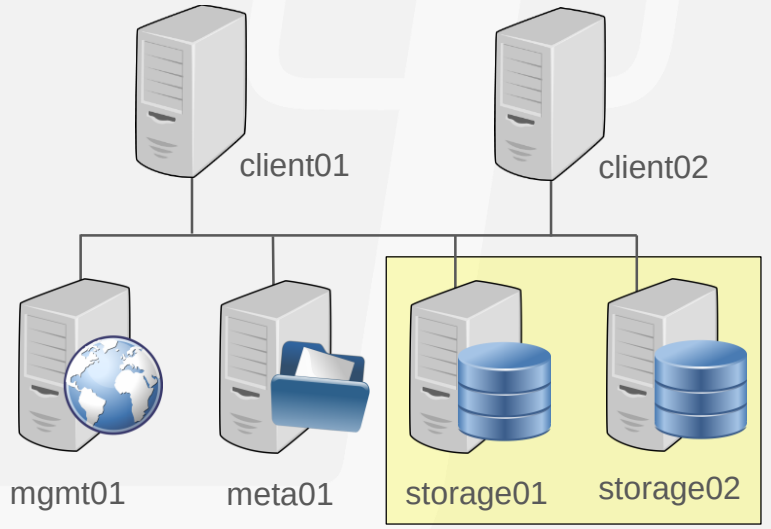
- Application can set pattern via API



Getting Server Statistics

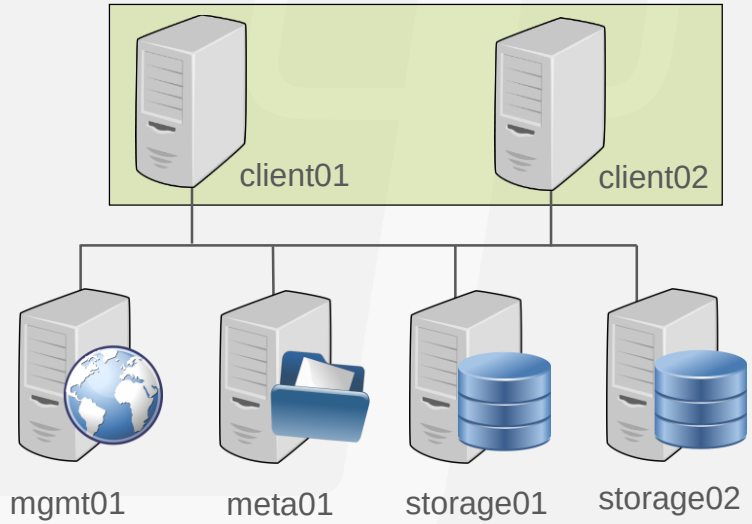
```
client01:~ # beegfs-ctl --serverstats --perserver --interval=1 --nodetype=storage

=== time index: 1412011239 (values show last second only)

nodeID write_KiB  read_KiB   reqs   qlen bsy
     1   1436430      5633  18098      0   0
     2   1536430     14430  18386      0   0

=== time index: 1412011240 (values show last second only)

nodeID write_KiB  read_KiB   reqs   qlen bsy
     1   1404430      9633  15613      0   0
     2   1336330     12403  17274      0   0
...
```





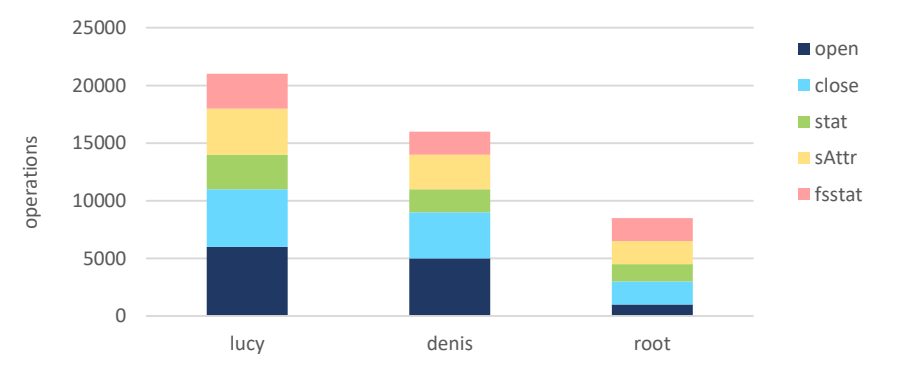
Getting Client Statistics

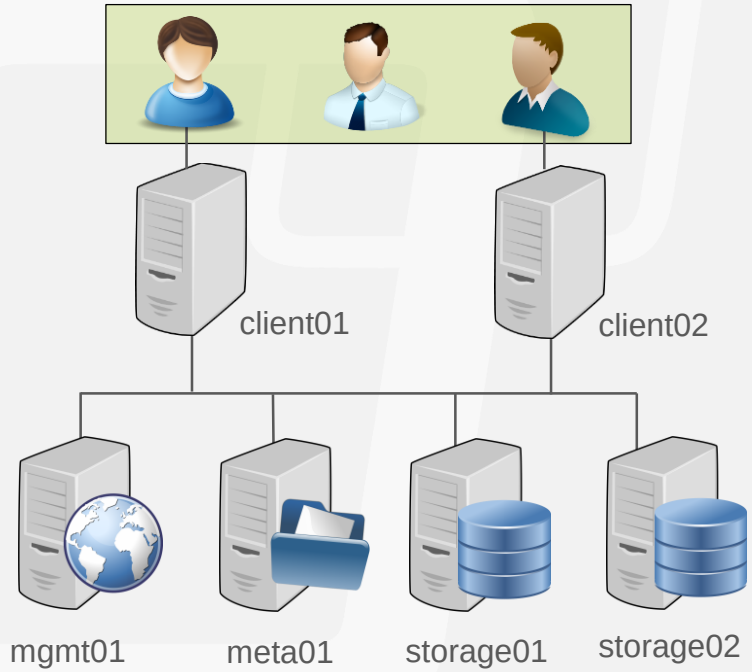
```
client01:~ # beegfs-ctl --clientstats --names --interval=5 nodetype=storage

====== 5 s ======
Sum       22967 [sum] 4720 [close]  65 [sAttr] 5706 [open] 2456 [stat] ...
client01   2847 [sum]  700 [close] 700 [open]  1447 [stat] ...
client02   2847 [sum]  700 [close] 700 [open]  1447 [stat] ...
meta01     1060 [sum] 1060 [close] ...
...
====== 10 s ======
Sum:      28138 [sum] 6511 [close] 1800 [open]  1804 [stat] 4 [sAttr] ...
client01   5803 [sum] 1403 [close] 1400 [open] ...
client02   5803 [sum] 1403 [close] 1400 [open] ...
meta01     2454 [sum] 2454 [close] ...
...
```



Getting User Statistics

```
client01:~ # beegfs-ctl --userstats --names --interval=5 --nodetype=meta

====== 5 s ======
Sum:          60 [sum] 10 [close]  10 [open]   1 [sAttr]  6 [unlnk] ...
lucy          36 [sum]  3 [close]   3 [open]   4 [stat]   1 [unlnk] ...
root          12 [sum]  6 [stat]    4 [unlnk] ...
denis          8 [sum]  1 [sAttr]   3 [stat] ...

====== 10 s ======
Sum:        2314 [sum] 24 [close]   1 [create] ...
lucy        2029 [sum] 23 [close]  23 [open] ...
denis         16 [sum]  1 [close]   1 [create] ...
root           8 [sum]  4 [stat] ...
```

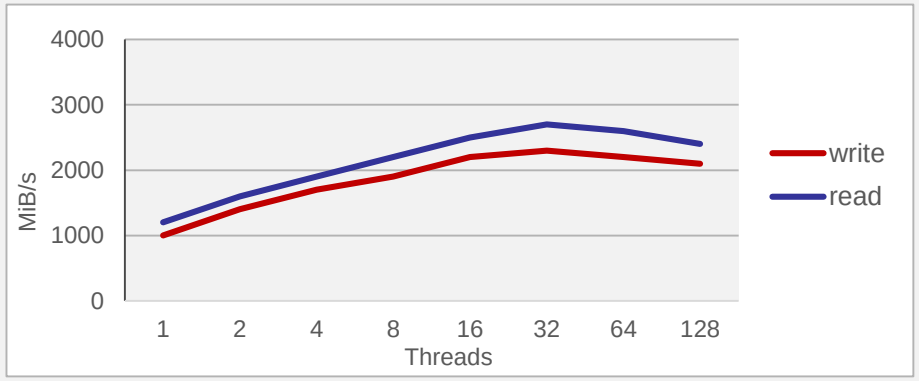


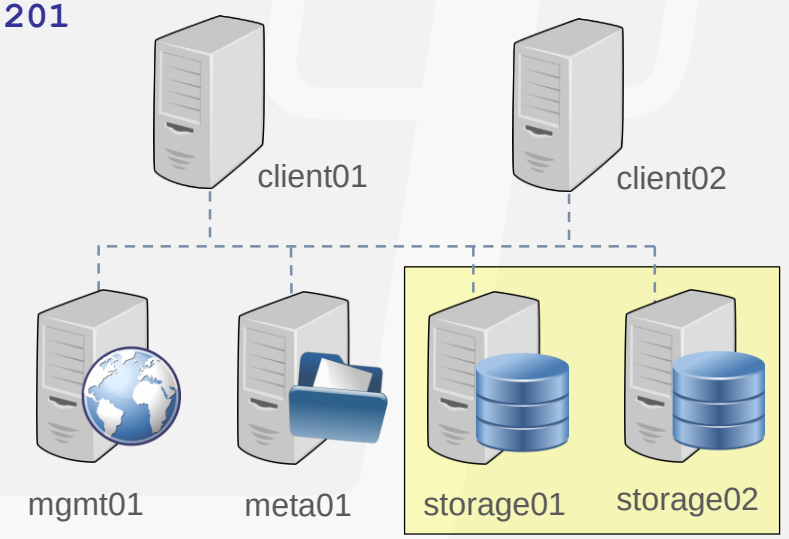


Built-in benchmark tool

- StorageBench

```
client01:~ # beegfs-ctl --storagebench --alltargets --write --blocksize=512K --size=320G --threads=4
client01:~ # beegfs-ctl --storagebench --alltargets --read --blocksize=512K --size=320G --threads=4

client01:~ # beegfs-ctl --storagebench --alltargets --status
Server benchmark status: Running:    4

Read benchmark results:
Min throughput:       1107 MiB/s nodeID: storage01 [ID: 1], targetID: 101
Max throughput:       1462 MiB/s nodeID: storage02 [ID: 2], targetID: 201
Avg throughput:       1283 MiB/s
Aggregate throughput: 2590 MiB/s
```

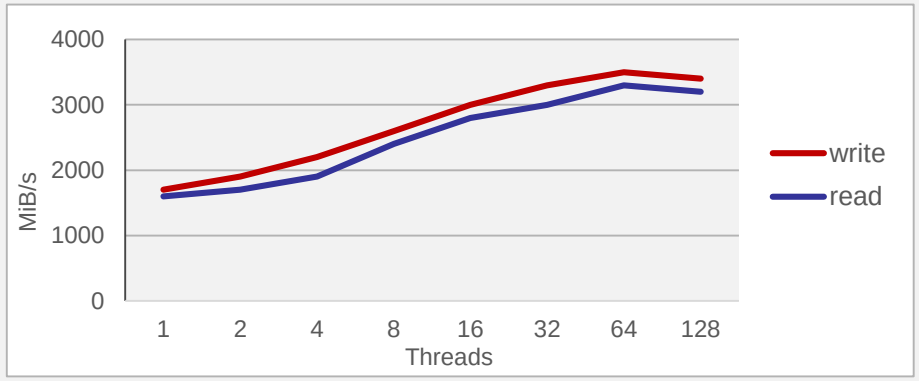


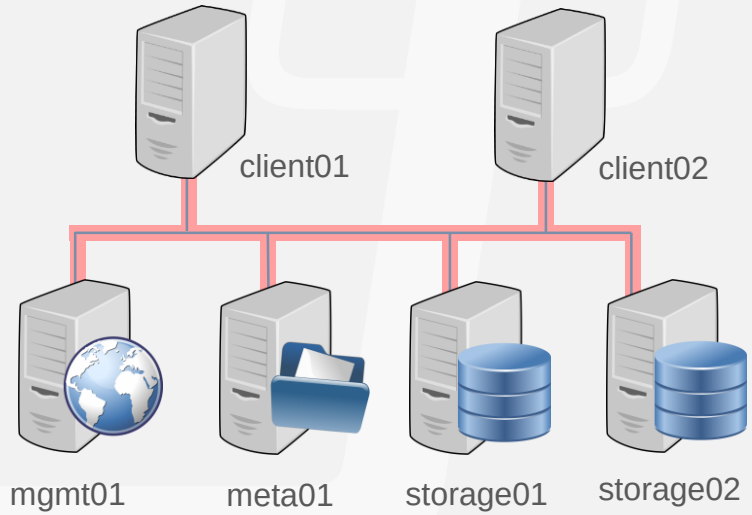


Built-in benchmark tool

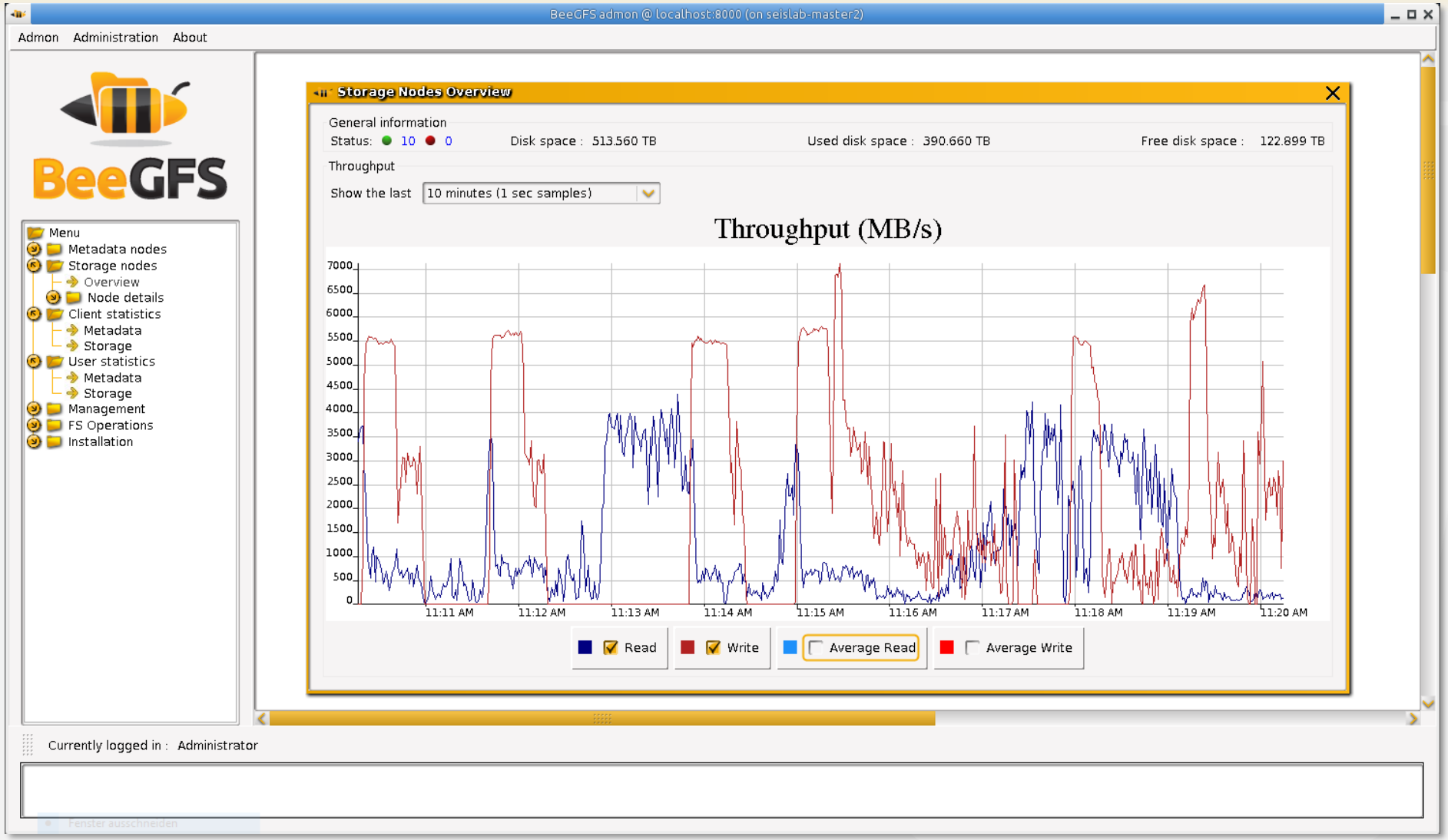
- NetBench mode
  - Network streaming throughput
  - Roughly equivalent to writing to a RAM drive

```
client01:~ # $ echo 1 > /proc/fs/beegfs/<clientID>/netbench_modes
...
client01:~ # $ echo 0 > /proc/fs/beegfs/<clientID>/netbench_modes
```





Admon GUI


BeeGFS admon @ localhost:8000 (on seislab-master2)
Admon
Administration
About
BeeGFS
Menu
Metadata nodes
Storage nodes
Overview
Node details
Client statistics
Metadata
Storage
User statistics
Metadata
Storage
Management
FS Operations
Installation
Storage Nodes Overview
General information
Status: 10 0
Disk space : 513.560 TB
Used disk space : 390.660 TB
Free disk space : 122.899 TB
Throughput
Show the last
10 minutes (1 sec samples)
Throughput (MB/s)
7000
6500
6000
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
11:11 AM
11:12 AM
11:13 AM
11:14 AM
11:15 AM
11:16 AM
11:17 AM
11:18 AM
11:19 AM
11:20 AM
Read
Write
Average Read
Average Write
Currently logged in : Administrator

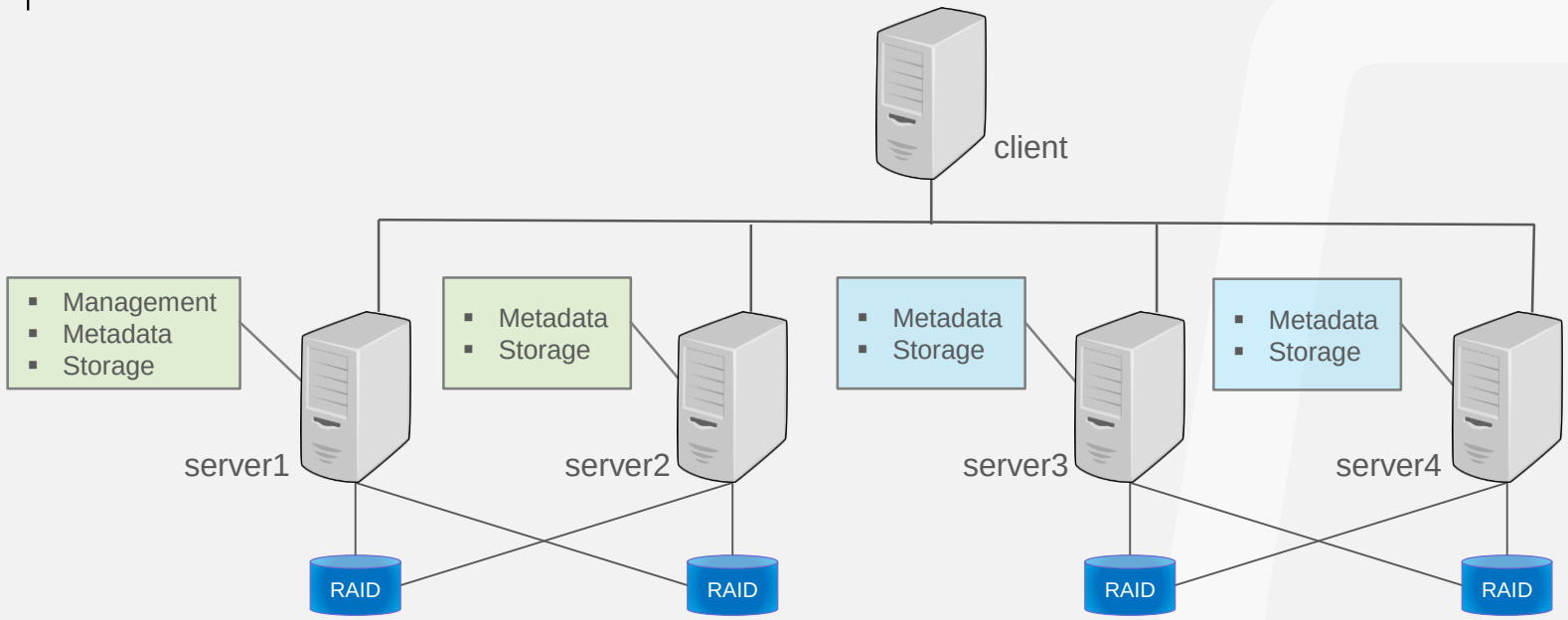
High Availability

Shared Storage

Buddy Mirroring

High Availability - Shared Storage

- Pacemaker Resource Management
  - BeeGFS services
  - Virtual IP addresses
  - Mount points
- Corosync
- STONITH – Resource Fencing Technique
- BeeGFS HA scripts



High Availability - Shared Storage

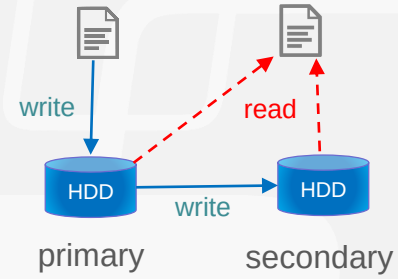
- No additional storage capacity needed

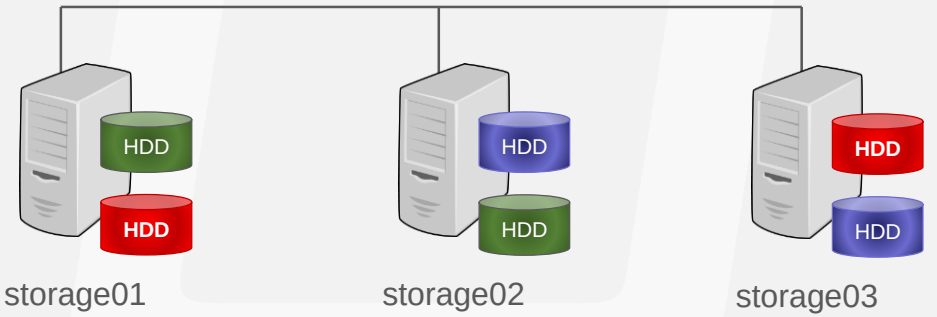
- Expensive storage components needed
- 3rd party software components needed
- Complex to set up and maintain
- Failover Risk
- No increased data safety

High Availability – Built-in Data replication

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

```
# beegfs-ctl –addmirrorgroup --automatic

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

# beegfs-ctl –listmirrorgroups

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

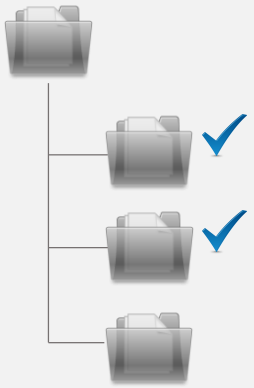


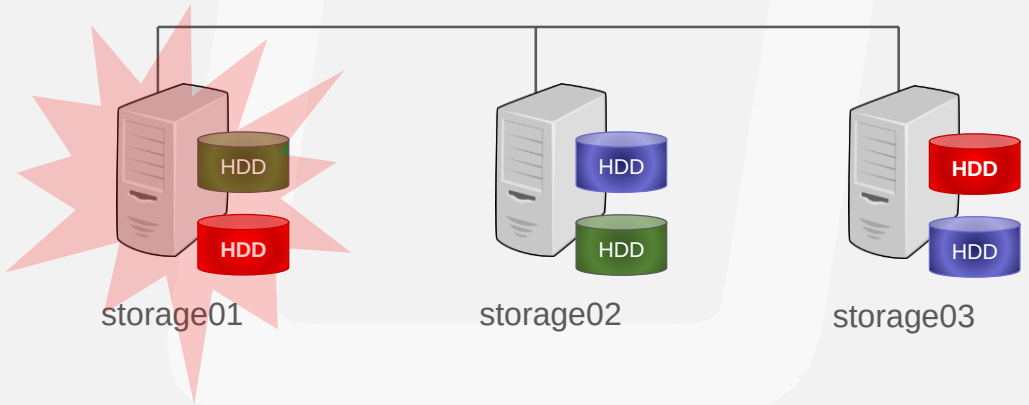


High Availability – Built-in Data replication

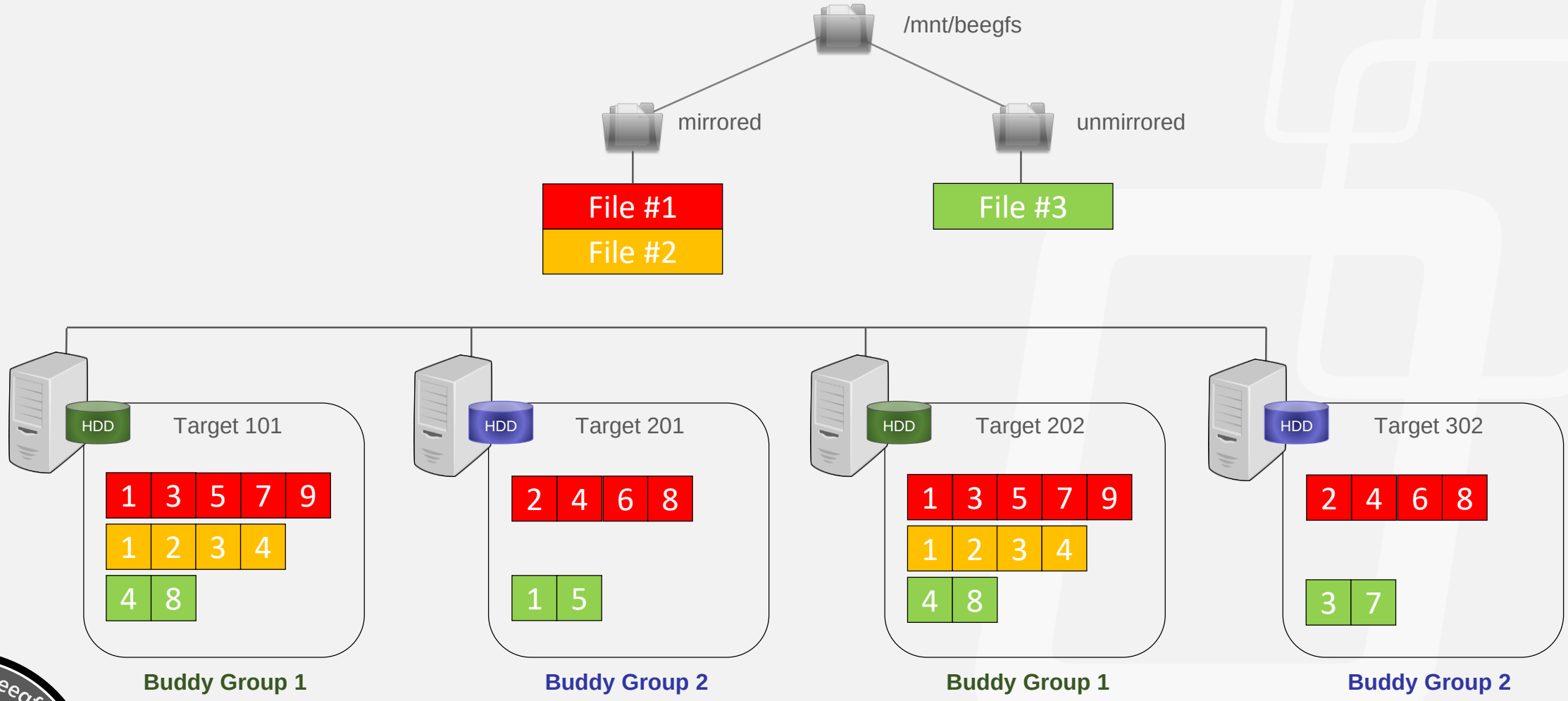
- Buddy Group Mirroring
  - Internal failover mechanism
    - In case primary is unreachable or fails, an automatic switch is performed
    - Self-healing (differential rebuild) when buddy comes back
  - Enabled globally or on a per-directory basis

```
# beegfs-ctl --setpattern –buddygroup --chunksize=1m
             --numtargets=4 /mnt/beegfs/mirrored
```





Buddy Mirroring per Directory



High Availability – Built-in Replication

- Flexible (replication configurable per-directory)
- Easy to scale and extend
- No 3rd party tools for monitoring and failover functionality
- Any storage backend can be used
- Additional data safety

- Overhead in storage capacity
- Write penalty for replicated data

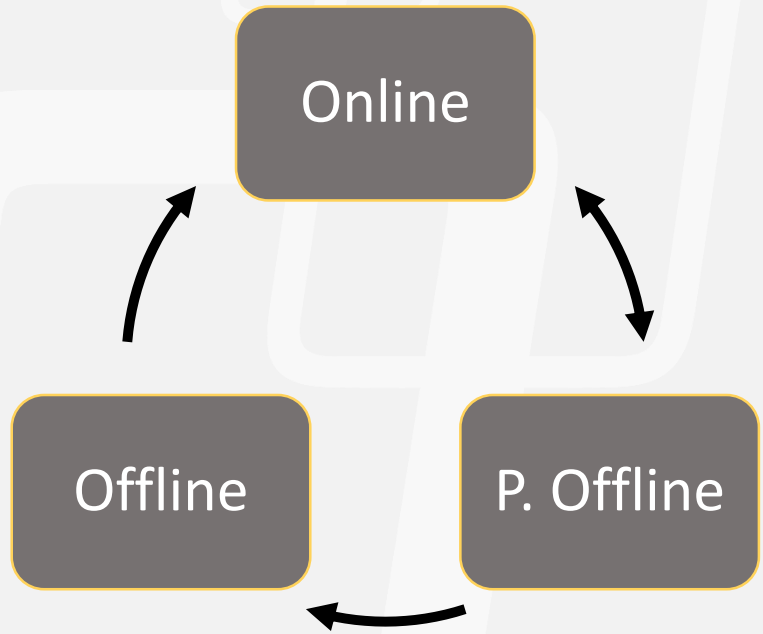
Target States

- Two different states for targets
  - Consistency state
  - Reachability state
- Internally used for several optimizations
- Important for HA

```
client01:~ # beegfs-ctl --listtargets --state

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
```

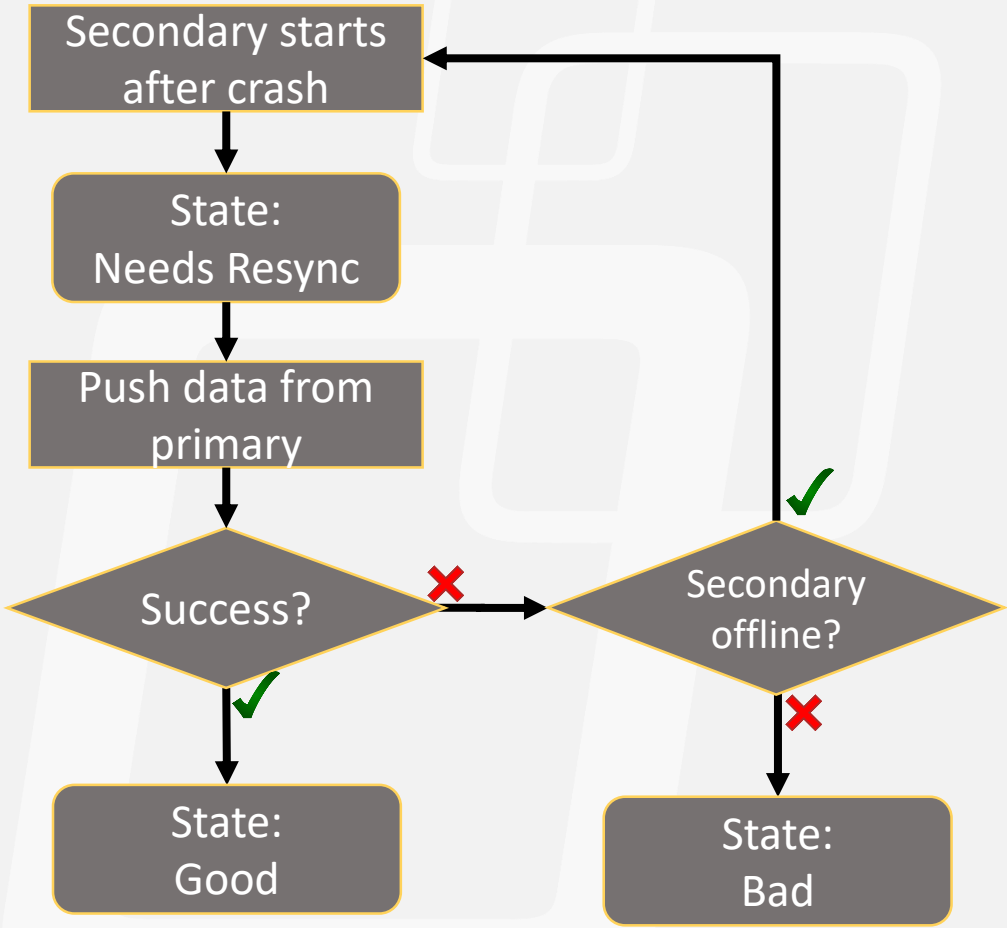
Reachability State

- Online
  - The target is reachable and fully usable by clients
- Probably Offline
  - The target might be offline
  - Mirrored files on this target may not be accessed
  - Non-mirrored files can be attempted (but may fail)
  - intermediate state avoid races and split-brain situations
- Offline
  - If target is part of a buddy group, try a switchover



Consistency State

- Good
  - Target may be used without limitations
- Needs Resync / Resyncing
  - Target needs a resync (or resync in progress)
  - Only valid for secondary targets
  - Clients may still access non-mirrored files
- Bad
  - A resync failed
  - Needs manual intervention

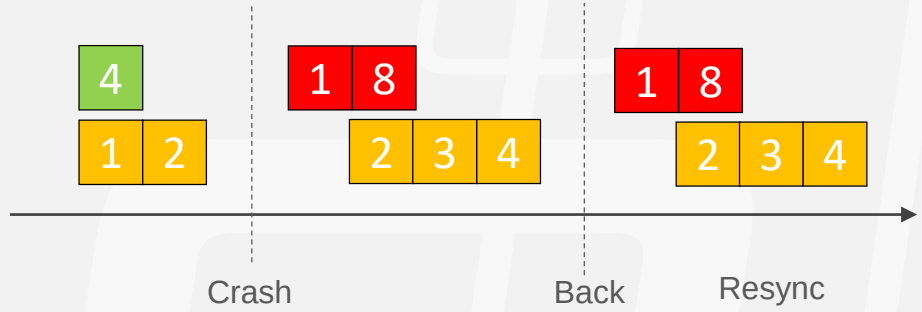


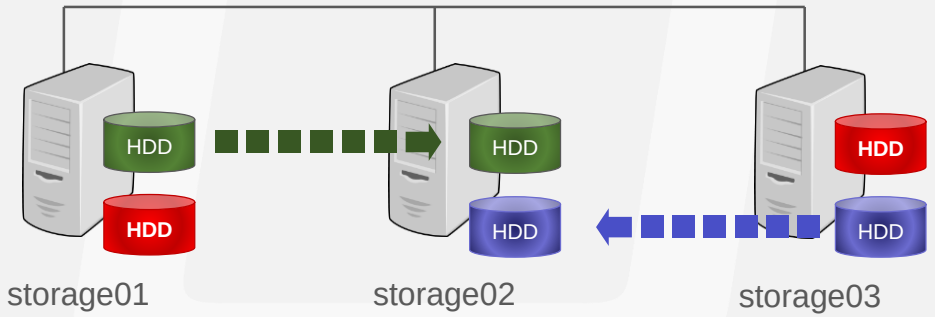
Resync

- Completely transparent & automatic in case 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. for 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, some new arguments

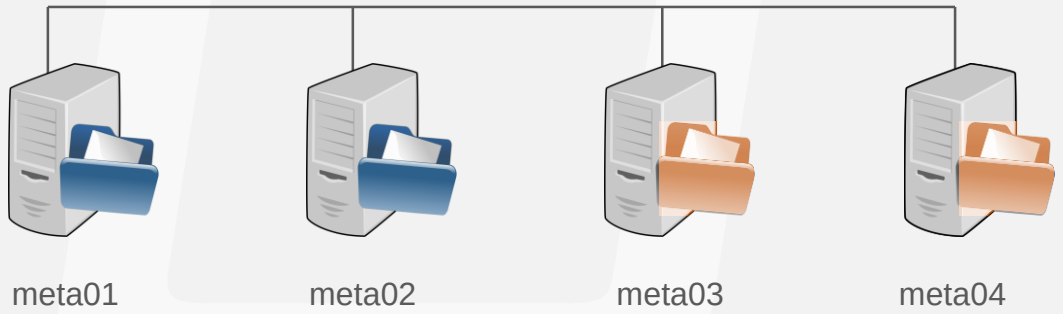
```
# beegfs-ctl --addmirrorgroup --nodetype=metadata
             --groupid=1 --primary=1 --secondary=2

# beegfs-ctl --listmirrorgroups

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

  - Same consistency and reachability states
  - Directories can have mirroring turned off

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

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

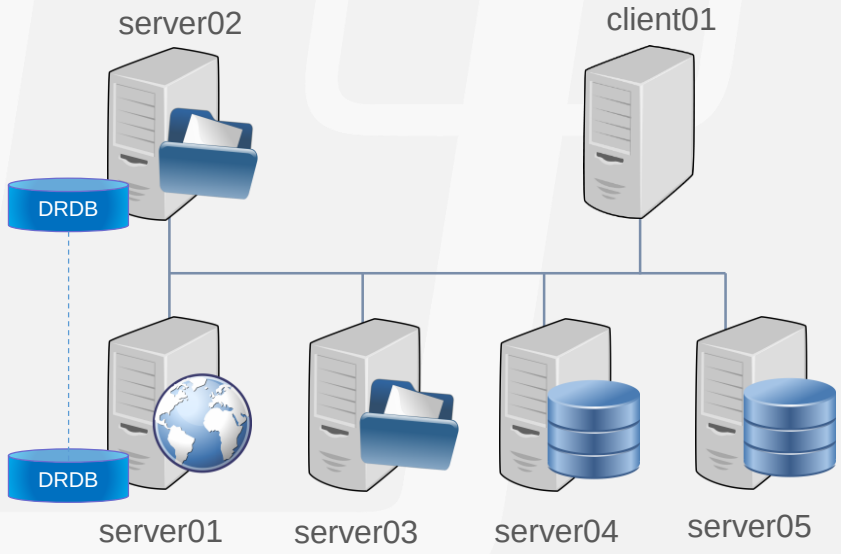
```
# beegfs-ctl --mirrormd
```



Management Mirroring

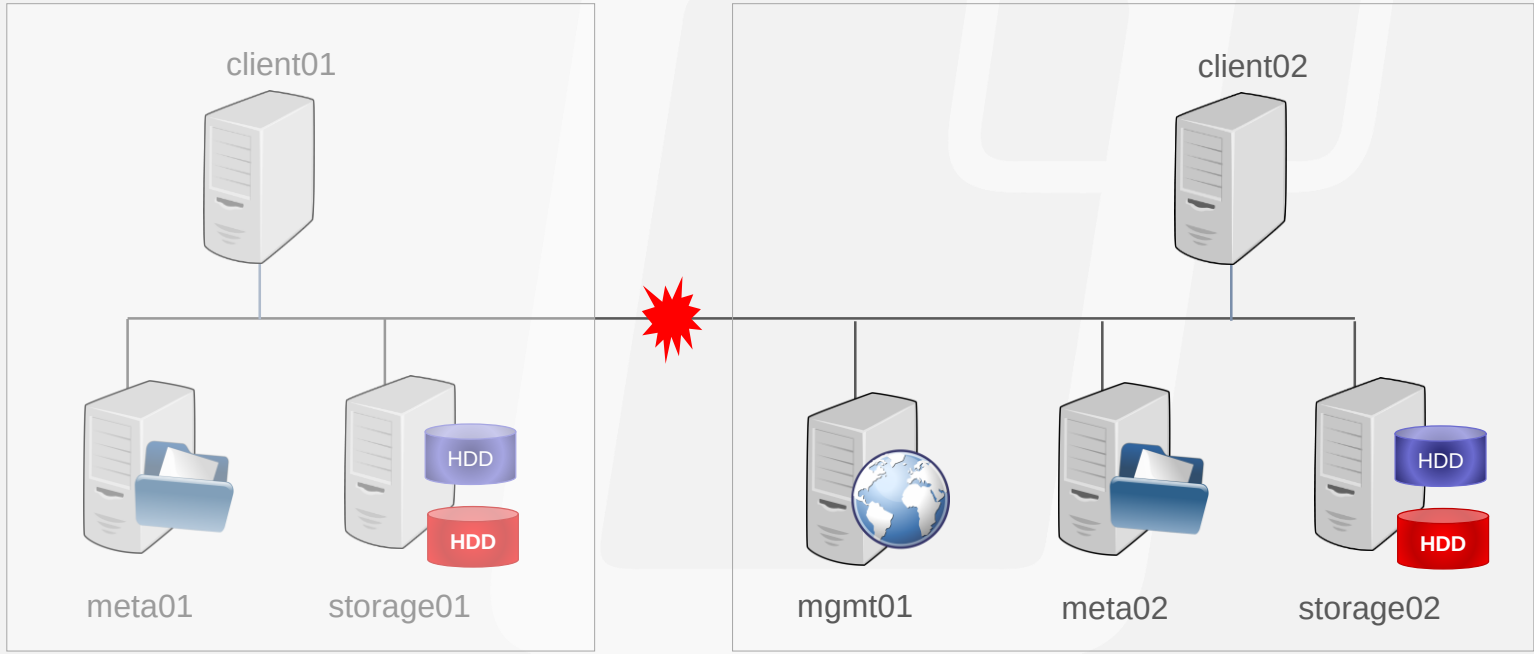
- Currently under development
  - Expected to be released later this year

- Alternative solution
  - DRDB volume for the management target
    - The management service is not performance critical
  - Tool like Pacemaker to start the management service and virtual IP on another machine when it fails



Split Brain

- Network split leads to a system partition
  - Nodes on the same partition as the management service continue working normally
  - Buddy mirroring failover may occur
  - Nodes on the other partition stall and resume working after connection is reestablished
  - Data is resynced

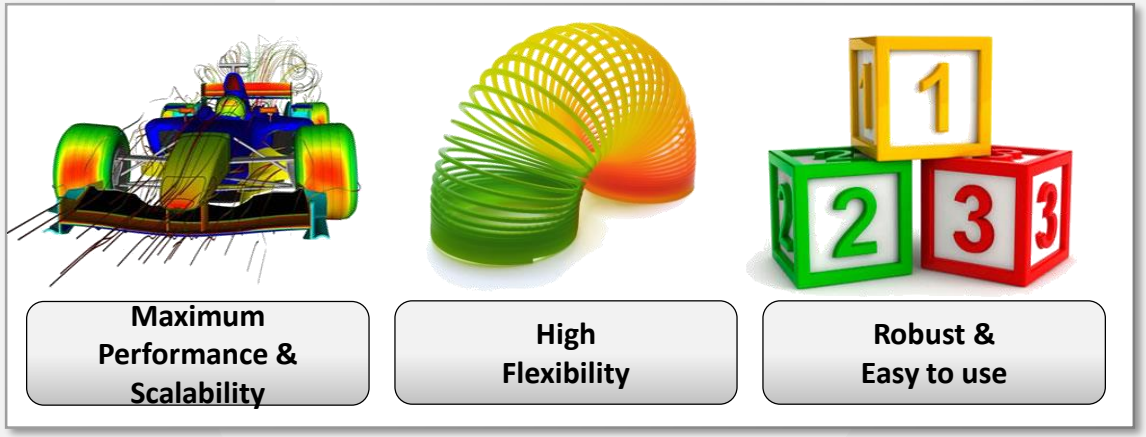


Conclusion

Conclusion

- Easy to use and robust parallel file system
- Excellent performance based on highly scalable architecture

- Flexible high availability solutions
  - Shared storage
  - Buddy Mirror group



Why do people go with BeeGFS?

"We are extremely happy with our 3.1PB BeeGFS on 30 servers – it runs rock-solid."
- Bioinformatics Aarhus, Denmark

FRED HUTCH
CURES START HERE™

"We're now at one year of uptime without a single hitch of the BeeGFS."
- Fred Hutchinson Medical Institute, USA

"After many unplanned downtimes with our previous parallel FS, we moved to BeeGFS more than 2 years ago. Since then we had no unplanned downtimes anymore."
- University of Halle, Germany

"We've seen BeeGFS performing very well and also saw its easy and robustness – which we did not see in many other parallel file systems."
- ClusterVision

Questions? / Keep in touch

- **Web** www.beegfs.com
  www.thinkparq.com
- **Mail** sales@thinkparq.com
  support@beegfs.com
- **Twitter** www.twitter.com/BeeGFS
- **Newsletter** www.beegfs.com/news