Hands on BeeGFS

Table of Contents

Main objectives

There's three parts in the practical session:

Practical Work 1

- Create an all-in-one beegfs_server_aio:
  - 4x CPUs
  - 2048MB RAM
  - 10GB metadata disk
  - 200GB storage disk
- Create a beegfs_client_1:
  - 1x CPU
  - 1024MB RAM

Practical Work 2

- Add a new storage jbod to beegfs_server_aio:
  - 200GB storage disk
- Create three more beegfs_client_2, beegfs_client_3, beegfs_client_4:
  - 1x CPU
  - 1024MB RAM

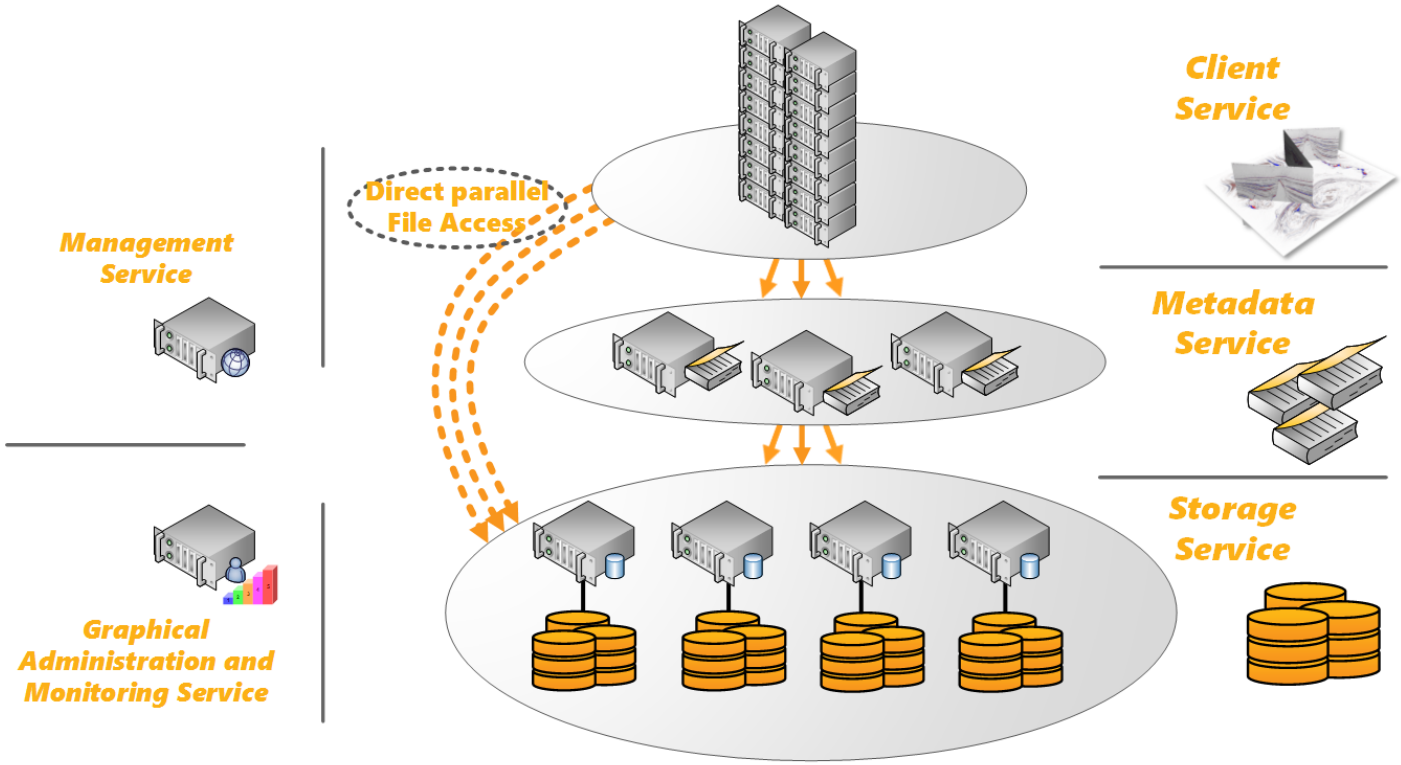
Practical Work 3

- Perform built-benchmarks
- Perform IOR banchmark

BeeGFS Concepts

Four main beegfs services:

- beegfs-mgmtd
- beegfs-meta
- beegfs-storage
- beegfs-client



Source of information at the BeeGFS website:

- Installation guide (http://www.beegfs.com/wiki/ManualInstallation)
- Example setup (http://www.beegfs.com/wiki/ManualInstallWalkThrough)

Useful application notes:

- Introduction to BeeGFS (http://www.beegfs.com/docs/whitepapers/Introduction_to_BeeGFS_by_ThinkParQ.pdf)
- Picking the right number of targets per storage server for BeeGFS (http://www.beegfs.com/docs/whitepapers/Picking_the_right_Number_of_Targets_per_Server_for_BeeGFS_by_ThinkParQ.pdf)
- Metadata Performance Evaluation of BeeGFS (http://www.beegfs.com/docs/whitepapers/Metadata_Performance_Evaluation_of_BeeGFS_by_ThinkParQ.pdf)

Practical Work 1

OpenNebula VM Startup

Prerequisite

- Save your OpenNebula login/password

```
[user@hyper6 ~]$ cat .one/one_auth
<opennebula_user>:<opennebula_password>
```

```
[user@hyper6 ~]$ vi .bashrc
export ONE_AUTH=~/.one/one_auth
```

- Generating public/private dsa key pair for VM login

```
[user@hyper6 ~]$ ssh-keygen
```

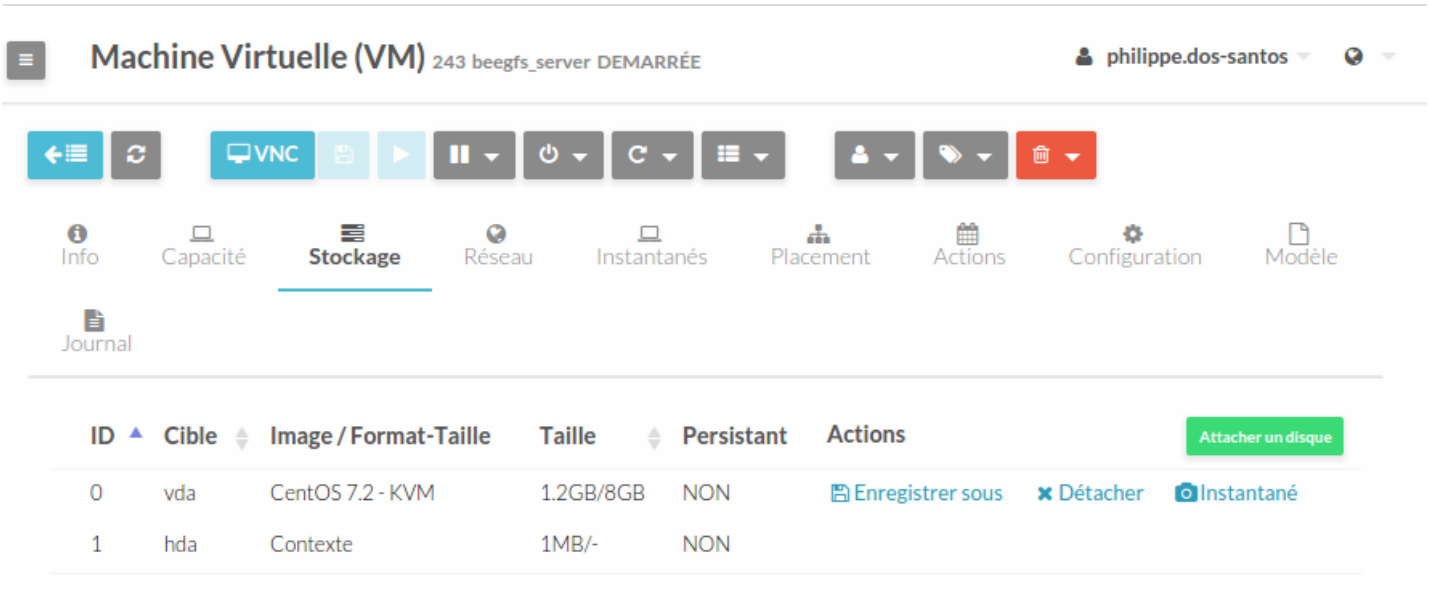
VM creation

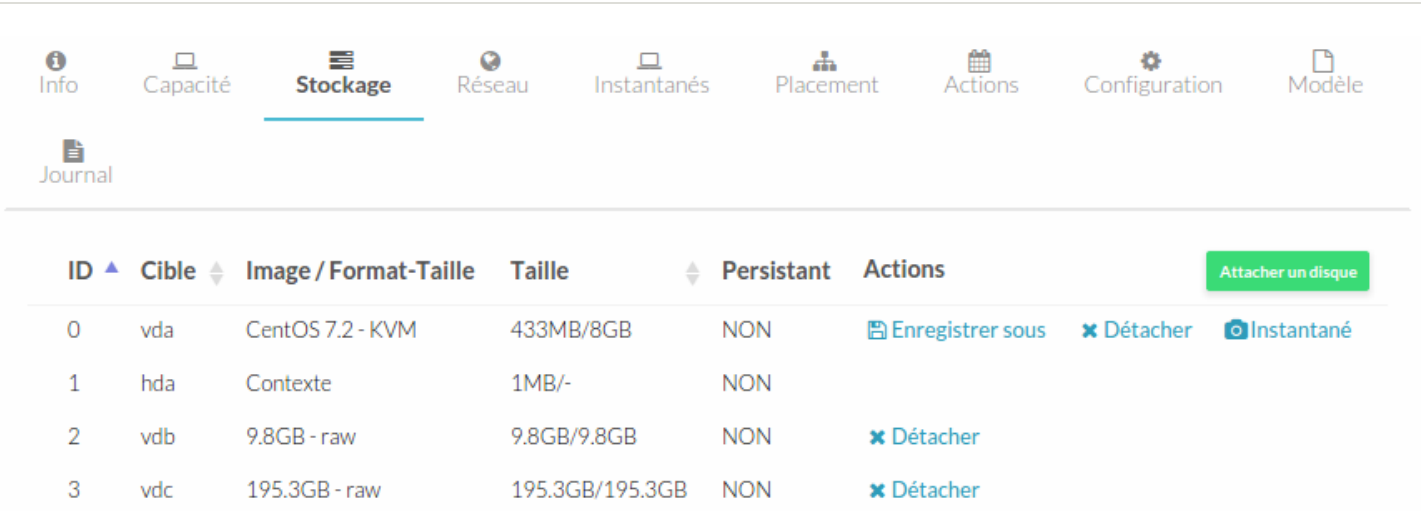
BeeGFS server

- Instantiate the all-in-one BeeGFS server

```
[user@hyper6 ~]$ onetemplate instantiate 3 --name beegfs_server_aio --cpu 4 --memory 2048 --nic
'oneadmin[PRIVATE_666]' --net_context --ssh '/home/user/.ssh/id_rsa.pub'
```

- Add metadata and storage disk





Log into the BeeGFS server

- Log in to the VM

```
[user@hyper6 ~]$ onevm list --list IP,USER,NAME,STAT --filter USER=$USER
IP              USER     NAME            STAT
10.100.0.X     user     beegfs_server_a   runn

[user@hyper6 ~]$ ssh root@10.100.0.X
```

- Metadata disk

SSDs widely used, RAID1 for low latency
0.5% of total storage space
ext4 (fast handling of small files)
http://www.beegfs.com/wiki/MetaServerTuning

```
# mkfs.ext4 -i 2048 -I 1024 -b 4096 -J size=400 -Odir_index,filetype /dev/vdb
```

- Storage disk

RAID6 volumes, 12 disks per volume
usually XFS, but also others like ZFS and ext4
http://www.beegfs.com/wiki/StorageServerTuning

```
# mkfs.xfs /dev/vdc
```

All-in-one BeeGFS Installation

BeeGFS Packages

- First things first, update the OS

```
# yum -y update
```

- Download the BeeGFS respository file

```
# yum -y install wget

# wget -O /etc/yum.repos.d/beegfs-rhel7.repo http://www.beegfs.com/release/latest-stable/dists/beegfs-rhel7.repo
```

- Install BeeGFS packages (server and client)

```
# yum -y install beegfs-mgmtd beegfs-meta beegfs-storage beegfs-client beegfs-helperd beegfs-utils
```

Disable SELinux

- Disable SELinux and check

```
# sed -i 's/enforcing/disabled/g' /etc/selinux/config

# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing   <---
Mode from config file:          disabled    <---
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

- Reboot and check

```
# reboot

# sestatus
SELinux status:                 disabled
```

Basic Configuration

Management service

- Start **beegfs-mgmtd**

/opt/beegfs/sbin/beegfs-setup-mgmtd

- For example

```
# mkdir -p /beegfs/mgmtd

# /opt/beegfs/sbin/beegfs-setup-mgmtd -p /beegfs/mgmtd/
    Preparing storage directory: /beegfs/mgmtd/
     * Creating format.conf file...
    Updating config file: /etc/beegfs/beegfs-mgmtd.conf
     * Setting storage directory in config file...
     * Disabling usage of uninitialized storage directory in config file...
    All done.

# systemctl start beegfs-mgmtd
# systemctl status beegfs-mgmtd
```

- Configuration file

```
# file /etc/beegfs/beegfs-mgmtd.conf
/etc/beegfs/beegfs-mgmtd.conf: ASCII text
```

Metadata service

- Start **beegfs-meta**

/opt/beegfs/sbin/beegfs-setup-meta

- Mount metadata disk

```
# mkdir /beegfs/meta
# mount /dev/vdb /beegfs/meta/
# mount
# vi /etc/fstab
...
/dev/vdb    /beegfs/meta    ext4    defaults    0   0
```

- Configure and start

```
# /opt/beegfs/sbin/beegfs-setup-meta -p /beegfs/meta/ -m 10.100.0.X
    Preparing storage directory: /beegfs/meta/
     * Creating format.conf file...
    Updating config file: /etc/beegfs/beegfs-meta.conf
     * Setting management host: 10.100.0.X
     * Setting storage directory in config file...
     * Disabling usage of uninitialized storage directory in config file...
     * Setting usage of extended attributes to: true
    All done.

# systemctl start beegfs-meta
# systemctl status beegfs-meta
```

- Configuration file

```
# file /etc/beegfs/beegfs-meta.conf
/etc/beegfs/beegfs-meta.conf: ASCII text
```

Storage service

- Start **beegfs-storage**

/opt/beegfs/sbin/beegfs-setup-storage

- Mount storage disk

```
# mkdir /beegfs/storage
# mount /dev/vdc /beegfs/storage/
# mount
# vi /etc/fstab
...
/dev/vdc    /beegfs/storage xfs     defaults    0   0
```

- Configure and start

```
# /opt/beegfs/sbin/beegfs-setup-storage -p /beegfs/storage/ -m 10.100.0.X
Preparing storage target directory: /beegfs/storage/
 * Creating format.conf file...
 * Creating chunks directory...
 * Creating buddymir directory...
Updating config file: /etc/beegfs/beegfs-storage.conf
 * Setting management host: 10.100.0.X
 * Appending to target directory list in config file...
 * Disabling usage of uninitialized storage targets in config file...
All done.


# systemctl start beegfs-storage
# systemctl status beegfs-storage
```

- Configuration file

```
# file /etc/beegfs/beegfs-storage.conf
/etc/beegfs/beegfs-storage.conf: ASCII text
```

Client service

- Start **beegfs-client**

/opt/beegfs/sbin/beegfs-setup-client

- For example

```
# /opt/beegfs/sbin/beegfs-setup-client -m 10.100.0.X
Updating config file: /etc/beegfs/beegfs-client.conf
 * Setting management host: 10.100.0.X
All done.

# systemctl start beegfs-client
Job for beegfs-client.service failed because the control process exited with error code. See
"systemctl status beegfs-client.service" and "journalctl -xe" for details.

# journalctl --unit beegfs-client
systemd[1]: Starting Start BeeGFS Client...
beegfs-client[16857]: Starting BeeGFS Client:
beegfs-client[16857]: - Loading BeeGFS modules
beegfs-client[16857]: - BeeGFS module autobuild
beegfs-client[16857]: Makefile:157: *** Linux kernel build directory not found. Please check if the
kernel module development packages are installed for the current kernel version. (RHEL: kernel-devel;
SLES: linux-kernel-headers, kernel-source; Debian: linux-headers).
systemd[1]: beegfs-client.service: main process exited, code=exited, status=1/FAILURE
systemd[1]: Failed to start Start BeeGFS Client.
systemd[1]: Unit beegfs-client.service entered failed state.
systemd[1]: beegfs-client.service failed.
beegfs-client[16857]: make: *** [auto_rebuild] Error 2
```

What's wrong?

- beegfs kernel module is missing

lsmod
/etc/init.d/beegfs-client

- Install what's needed to compile the beegfs kernel module

```
# lsmod | grep beegfs

# yum -y install kernel-devel gcc

# systemctl start beegfs-client
# systemctl status beegfs-client
...
systemd[1]: Starting Start BeeGFS Client...
beegfs-client[31819]: Starting BeeGFS Client:
beegfs-client[31819]: - Loading BeeGFS modules
beegfs-client[31819]: - Mounting directories from /etc/beegfs/beegfs-mounts.conf
systemd[1]: Started Start BeeGFS Client.

# lsmod | grep beegfs
beegfs                415176  1
```

- Configuration files

```
# cat /etc/beegfs/beegfs-mounts.conf
/mnt/beegfs /etc/beegfs/beegfs-client.conf

# file /etc/beegfs/beegfs-client.conf
/etc/beegfs/beegfs-client.conf: ASCII text
```

Beegfs module with InfiniBand support

```
# vi /etc/beegfs/beegfs-client-autobuild.conf
...
buildArgs=-j8 BEEGFS_OPENTK_IBVERBS=1
...

# beegfs-setup-rdma
Running Infiniband auto-detection...

Setting symlink in /opt/beegfs/lib: libbeegfs-opentk.so -> libbeegfs-opentk-enabledIB.so

# /etc/init.d/beegfs-client rebuild
- BeeGFS module autobuild
Building beegfs client module
```

Configuration and log Files

Configuration Files

- All configuration files are in /etc/beegfs

```
# ls -l /etc/beegfs/beegfs*.conf
-rw-r--r--. 1 root root  1725  2 déc.  12:24 beegfs-client-autobuild.conf
-rw-r--r--  1 root root 15054 10 déc.  09:13 beegfs-client.conf
-rw-r--r--. 1 root root  2298  2 déc.  07:33 beegfs-helperd.conf
-rw-r--r--. 1 root root   557  2 déc.  12:24 beegfs-libopentk.conf
-rw-r--r--  1 root root 12404 10 déc.  09:40 beegfs-meta.conf
-rw-r--r--  1 root root 12107 10 déc.  08:49 beegfs-mgmtd.conf
-rw-r--r--. 1 root root    43  2 déc.  12:24 beegfs-mounts.conf
-rw-r--r--  1 root root 12696 10 déc.  09:08 beegfs-storage.conf
```

- Human readable plain ASCII text and well documented

Log Files

- Each log file is located in /var/log

```
# ls -l /var/log/beegfs*.log
-rw-r--r-- 1 root root 7714 10 déc.  09:41 /var/log/beegfs-client.log
-rw-r--r-- 1 root root 1317 10 déc.  09:40 /var/log/beegfs-meta.log
-rw-r--r-- 1 root root 2528 10 déc.  09:41 /var/log/beegfs-mgmtd.log
-rw-r--r-- 1 root root 1396 10 déc.  09:49 /var/log/beegfs-storage.log
```

Systemd and journalctl

- Easier now to use the journalctl commande line

journalctl --unit=name
journalctl --unit=name1 --unit=name2
journalctl --unit=name --since=today
journalctl --unit=name -b 0
journalctl --unit=name -f
http://0pointer.de/blog/projects/journalctl.html

Check servers

Check reachability, connections and disk space

- Simple helpers

beegfs-check-servers
beegfs-net
beegfs-df

```
# beegfs-check-servers
Management
==========
localhost.localdomain [ID: 1]: reachable at 10.100.0.X:8008 (protocol: TCP)

Metadata
==========
localhost.localdomain [ID: 1]: reachable at 10.100.0.X:8005 (protocol: TCP)

Storage
==========
localhost.localdomain [ID: 1]: reachable at 10.100.0.X:8003 (protocol: TCP)


# beegfs-net

mgmt_nodes
=============
localhost.localdomain [ID: 1]
   Connections: TCP: 1 (10.100.0.X:8008);

meta_nodes
=============
localhost.localdomain [ID: 1]
   Connections: TCP: 1 (10.100.0.X:8005);

storage_nodes
=============
localhost.localdomain [ID: 1]
   Connections: TCP: 1 (10.100.0.X:8003);


# beegfs-df
METADATA SERVERS:
TargetID        Pool        Total         Free    %      ITotal       IFree    %
========        ====        =====         ====    =      ======       =====    =
       1         low       4.5GiB       4.2GiB  93%        4.9M        4.8M  99%

STORAGE TARGETS:
TargetID        Pool        Total         Free    %      ITotal       IFree    %
========        ====        =====         ====    =      ======       =====    =
       1         low     195.2GiB     195.2GiB 100%      195.3M      195.3M 100%
```

BeeGFS Client Installation

- Instantiate BeeGFS client

```
$ onetemplate instantiate 3 --name beegfs_client_1 --cpu 1 --memory 1024 --nic 'oneadmin[PRIVATE_666]'
--net_context --ssh '/home/user/.ssh/id_rsa.pub'
```

- Log in to the VM

```
[user@hyper6 ~]$ onevm list --list IP,USER,NAME,STAT --filter USER=$USER
IP              USER     NAME            STAT
10.100.0.X     user     beegfs_server_a   runn
10.100.0.Y     user     beegfs_client_1   runn

[user@hyper6 ~]$ ssh root@10.100.0.Y
```

BeeGFS Packages

- Update the OS

```
# yum -y update
```

- Download the BeeGFS respository file and install BeeGFS client packages

```
# yum -y install wget

# wget -O /etc/yum.repos.d/beegfs-rhel7.repo http://www.beegfs.com/release/latest-stable/dists/beegfs-rhel7.repo

# yum -y install beegfs-client beegfs-helperd beegfs-utils kernel-devel gcc
```

Disable SELinux

- Disable SELinux and reboot and check

```
# sed -i 's/enforcing/disabled/g' /etc/selinux/config

# reboot

[user@hyper6 ~]$ ssh root@10.100.0.Y

# sestatus
SELinux status:                 disabled
```

Basic Client Configuration

- Start beegfs-client

```
# /opt/beegfs/sbin/beegfs-setup-client -m 10.100.0.X
Updating config file: /etc/beegfs/beegfs-client.conf
 * Setting management host: 10.100.0.X
All done.

# systemctl start beegfs-helperd

# systemctl start beegfs-client
```

- Check mount point

```
# mount | grep beegfs
beegfs_nodev on /mnt/beegfs type beegfs (rw,relatime,cfgFile=/etc/beegfs/beegfs-client.conf,_netdev)
```

- Check space

```
# beegfs-df

METADATA SERVERS:
TargetID        Pool        Total         Free    %      ITotal       IFree    %
========        ====        =====         ====    =      ======       =====    =
       1         low       4.5GiB       4.2GiB  93%        4.9M        4.8M  99%

STORAGE TARGETS:
TargetID        Pool        Total         Free    %      ITotal       IFree    %
========        ====        =====         ====    =      ======       =====    =
       1         low     195.2GiB     195.2GiB 100%       97.7M       97.7M 100%
```

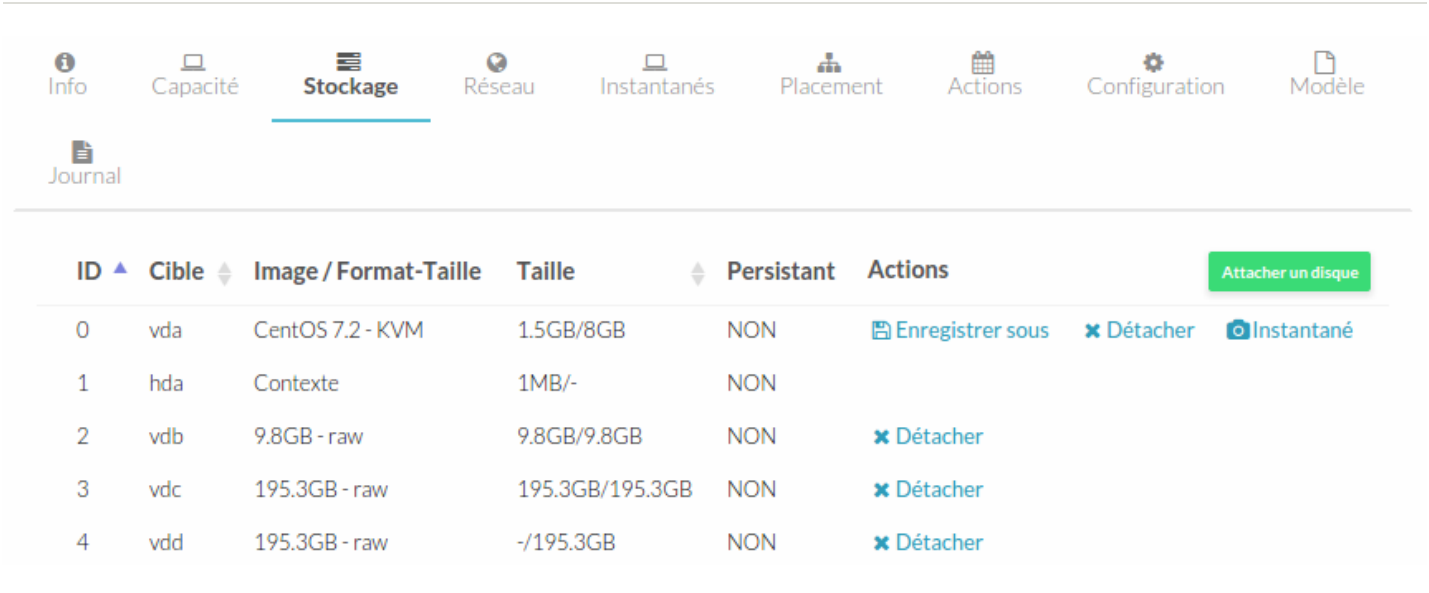
- Add file

```
# cd /mnt/beegfs/
# dd if=/dev/zero of=avant bs=1k count=1k
# ls /mnt/beegfs -l
total 1024
-rw-r--r-- 1 root root 1048576 14 déc.  06:09 avant
```

Practical Work 2

All-in-one BeeGFS server JBOD extension

- Add storage disk



Log into the BeeGFS server

- Log in to the VM

```
[user@hyper6 ~]$ onevm list --list IP,USER,NAME,STAT --filter USER=$USER
IP              USER     NAME            STAT
10.100.0.X     user     beegfs_server_a   runn
10.100.0.Y     user     beegfs_client_1   runn

[user@hyper6 ~]$ ssh root@10.100.0.X
```

- Format storage disk

```
# mkfs.xfs /dev/vdb

# mkdir -p /beegfs/storage_jbod
# mount /dev/vdb /beegfs/storage_jbod/
# mount
# vi /etc/fstab
...
/dev/vdb    /beegfs/storage xfs     defaults    0   0
```

Add new storage disk

- Stop BeeGFS services

```
# systemctl stop beegfs-client
# systemctl stop beegfs-storage
# systemctl stop beegfs-meta
```

- Identify current storage server and target ID

```
# cat /beegfs/storage/nodeNumID
1
# cat /beegfs/storage/targetNumID
1
```

- Initialize new target with new target ID and check config

```
# /opt/beegfs/sbin/beegfs-setup-storage -p /beegfs/storage_jbod/ -s 1 -i 2 -m 10.100.0.X
# vi /etc/beegfs/beegfs-storage.conf
...
storeStorageDirectory        = ,/beegfs/storage/ ,/beegfs/storage_jbod/
...

# cat /beegfs/storage_jbod/nodeNumID
1
# cat /beegfs/storage_jbod/targetNumID
2
```

- Start all BeeGFS services

```
# systemctl restart beegfs-mgmtd
# systemctl start beegfs-meta
# systemctl start beegfs-storage
# systemctl start beegfs-client
```

- Check space

```
# beegfs-df
METADATA SERVERS:
TargetID        Pool        Total         Free    %      ITotal       IFree    %
========        ====        =====         ====    =      ======       =====    =
       1         low       4.5GiB       4.2GiB  93%        4.9M        4.8M  99%

STORAGE TARGETS:
TargetID        Pool        Total         Free    %      ITotal       IFree    %
========        ====        =====         ====    =      ======       =====    =
       1         low     195.2GiB     195.2GiB 100%       97.7M       97.7M 100%
       2         low     195.2GiB     195.2GiB 100%       97.7M       97.7M 100%
```

- Check if file still there

```
# ls /mnt/beegfs -l
total 1024
-rw-r--r-- 1 root root 1048576 14 déc.  06:09 avant
```

Create three more BeeGFS clients

- beegfs_client_2, beegfs_client_3 and beegfs_client_4

```
$ onetemplate instantiate 3 --name beegfs_client_2 --cpu 1 --memory 1024 --nic 'oneadmin[PRIVATE_666]'
--net_context --ssh '/home/user/.ssh/id_rsa.pub

$ onetemplate instantiate 3 --name beegfs_client_3 --cpu 1 --memory 1024 --nic 'oneadmin[PRIVATE_666]'
--net_context --ssh '/home/user/.ssh/id_rsa.pub''

$ onetemplate instantiate 3 --name beegfs_client_4 --cpu 1 --memory 1024 --nic 'oneadmin[PRIVATE_666]'
--net_context --ssh '/home/user/.ssh/id_rsa.pub'
```

- Log in to each client, update, deactivate SELinux and install BeeGFS client packages

Practical Work 3

Built-in Benchmark Tools

beegfs-ctl --storagebench --help

Write

- Launch benchmark

```
# beegfs-ctl --storagebench --alltargets --write --blocksize=512K --size=100G --threads=4 --
mount=/mnt/beegfs/

Write storage benchmark was started.
You can query the status with the --status argument of beegfs-ctl.

Server benchmark status:
Running:      1
```

- Check status

```
# beegfs-ctl --storagebench --alltargets --status

Server benchmark status:
Running:      1

Write benchmark results:
Min throughput:             288254  KiB/s   nodeID: localhost.localdomain [ID: 1], targetID: 1
Max throughput:             360172  KiB/s   nodeID: localhost.localdomain [ID: 1], targetID: 2
Avg throughput:             324213  KiB/s
Aggregate throughput:       648426  KiB/s
```

- Stop benchmark

```
# beegfs-ctl --storagebench --alltargets --stop

Stopping storage benchmark. This will take a few moments.
You can query the status and the results with the --status argument of beegfs-ctl.

Server benchmark status:
Stopping:     1
```

Read

- Launch benchmark

```
# beegfs-ctl --storagebench --alltargets --read --blocksize=512K --size=100G --threads=4 --
mount=/mnt/beegfs/

Some errors occurred:
localhost.localdomain [ID: 1]: No (or not enough) data for read benchmark available.
For more details look into the storage server log file on the mentioned storage server.

Server benchmark status:
Error:        1
```

What's wrong?

beegfs-ctl --storagebench --help

- Read benchmark requires previous write benchmark

```
# beegfs-ctl --storagebench --help
...
    --write    Start a write benchmark.
    --read     Start a read benchmark (requires previous write benchmark)
...
```

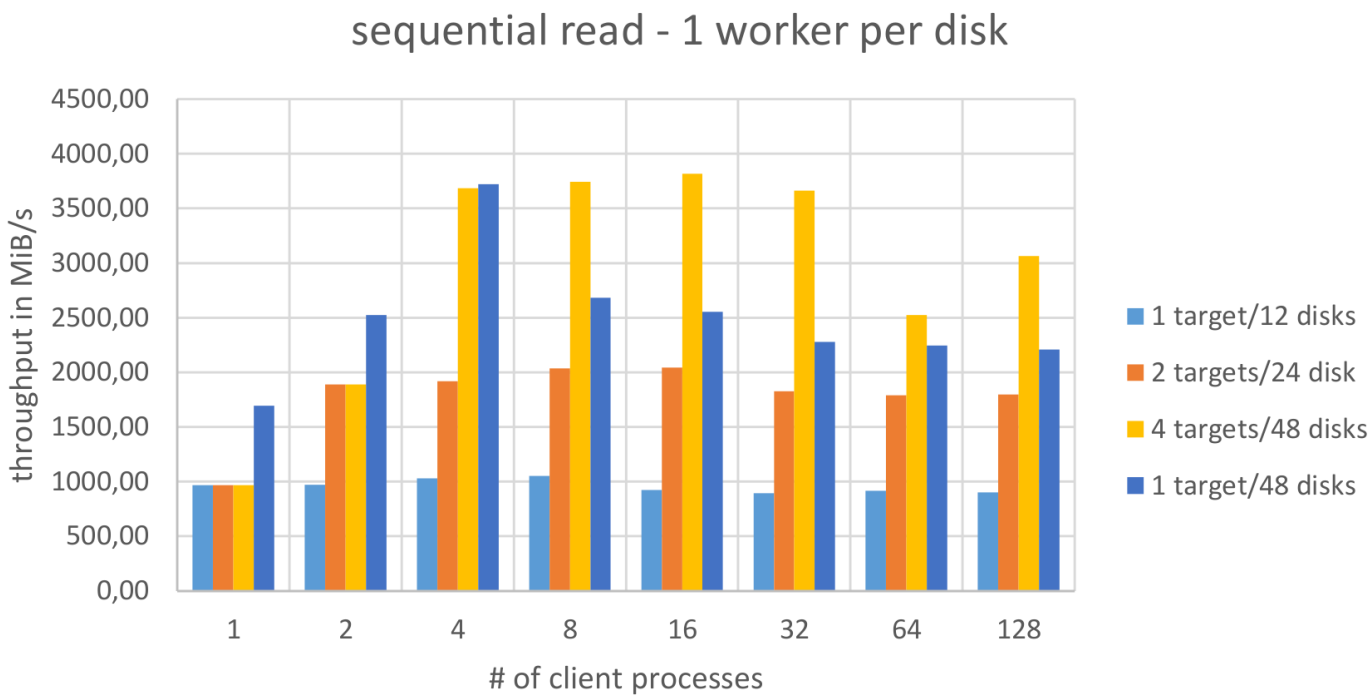
IOR Benchmark Tool

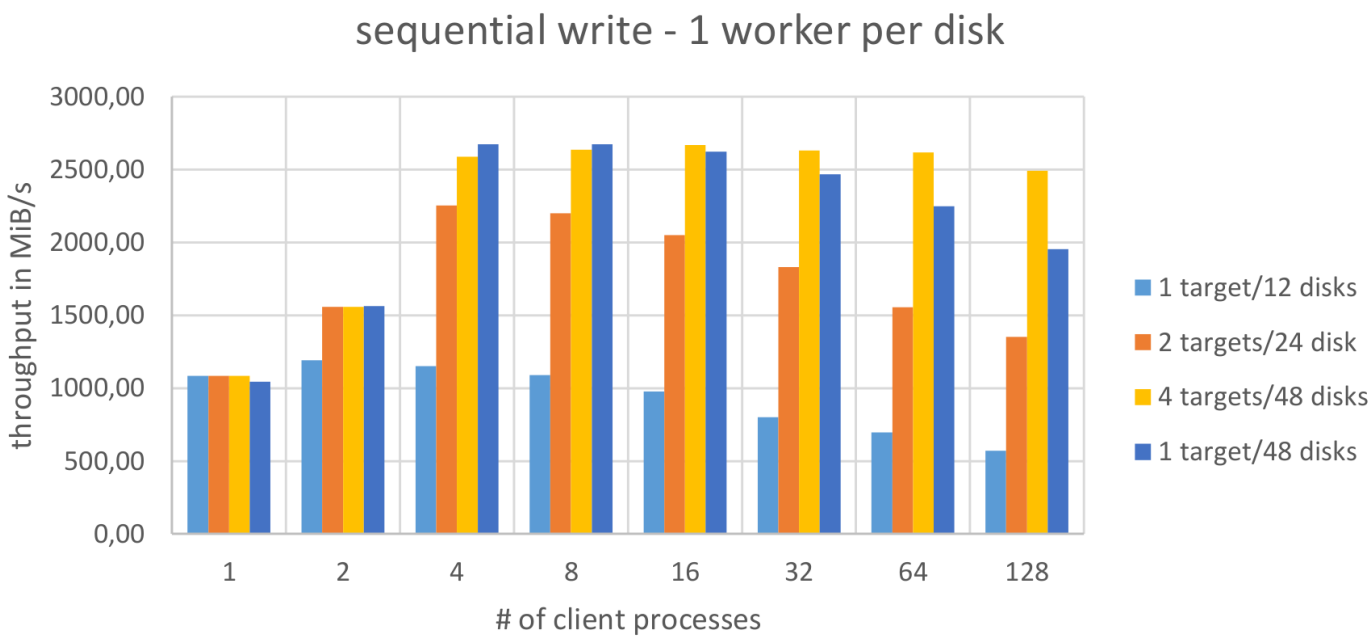
IOR Tool

"The IOR software is used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces."

From the whitepaper Picking the right number of targets per storage server for BeeGFS
(http://www.beegfs.com/docs/whitepapers/Picking_the_right_Number_of_Targets_per_Server_for_BeeGFS_by_ThinkParQ.pdf)

Assume one target is one disk: **i)** 1x target = 1x disk, **ii)** 2x targets = 2x disks, **iii)** 4x targets = 4x disks, **iv)** just for information, not useful.





Build IOR

- Download from: https://sourceforge.net/projects/ior-sio/

```
# yum -y install openmpi openmpi-devel
# exit

# ssh root@10.100.0.Y

# wget https://netix.dl.sourceforge.net/project/ior-sio/IOR%20latest/IOR-2.10.3/IOR-2.10.3.tgz
# tar xzvf IOR-2.10.3.tgz
# cd IOR

# module load mpi/openmpi-x86_64


# make
# cd src/C
```

Run on One Node

- Launch IOR

```
$ mpirun -np 1 IOR -r -w -t2m -F -g -b 100g -i 1 -o /mnt/beegfs/

IOR-2.10.3: MPI Coordinated Test of Parallel I/O

Run began: Wed Dec 14 09:54:54 2016
Command line used: IOR -r -w -t2m -F -g -b 100g -i 1 -o /mnt/beegfs/
Machine: Linux localhost.localdomain

Summary:
    api                = POSIX
    test filename      = /mnt/beegfs/
    access             = file-per-process
    ordering in a file = sequential offsets
    ordering inter file= no tasks offsets
    clients            = 1 (1 per node)
    repetitions        = 1
    xfersize           = 2 MiB
    blocksize          = 100 GiB
    aggregate filesize = 100 GiB
...
```

Run on Multiple Nodes

- Launch on two hosts

```
$ mpirun  -np 2 --hosts 10.100.0.Y,10.100.0.Z IOR -r -w -t2m -F -g -b 100g -i 1 -o /mnt/beegfs/
...
```

- Launch on four hosts

```
...
```

How about adding disks?

- Add two more disks and check performance with built-in benchmark and IOR benchmark

. . .

APPENDICES

General command line

- One command line control tool

beegfs-ctl

- beegfs-ctl help

```
# beegfs-ctl
BeeGFS Command-Line Control Tool (http://www.beegfs.com)
Version: 6.2

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.
 --setpattern (*)        => Set a new striping configuration.
 --mirrormd (*)          => Enable metadata mirroring.
 --find                  => Find files located on certain servers.
 --refreshentryinfo      => Refresh file system entry metadata.

 --createfile (*)        => Create a new file.
 --createdir (*)         => Create a new directory.
 --migrate               => Migrate files to other storage servers.
 --disposeunused (*)     => Purge remains of unlinked files.

 --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.
 --startresync (*)       => Start resync of a storage target.
 --resyncstats           => Get statistics on a storage target resync.

*) Marked modes require root privileges.

USAGE:
 This is the BeeGFS command-line control tool.

 Choose a control mode from the list above and use the parameter "--help" to
 show arguments and usage examples for that particular mode.

 Example: Show help for mode "--listnodes"
  $ beegfs-ctl --listnodes --help
```

- Listing registered client nodes

beegfs-ctl --listnodes --help

- Example

```
#  beegfs-ctl --listnodes --nodetype=client --details
7C62-584BCDC9-localhost.localdomain [ID: 4]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ens3(TCP)

Number of nodes: 1
```

Quota

Quota Setup

Follow these steps to implement BeeGFS quota

1. Enable quota tracking
2. Enable quota enforcement
3. Set/get quota limits

Enable Quota Tracking

- Shut down all BeeGFS services
- On storage target, enable quota on the underlying file system

man xfs_quota
http://www.beegfs.com/wiki/EnableQuota#oldinstall

```
# vi /etc/fstab
...
/dev/vdc        /beegfs/storage xfs     uqnoenforce,gqnoenforce         0       0
```

- Enable quota support in the config files of all clients

```
# vi /etc/beegfs/beegfs-client.conf
...
quotaEnabled = true
```

- Start all BeeGFS services

```
# reboot
```

- Run the beegfs-fsck tool to update the chunk files for quota support

```
# beegfs-fsck --enablequota
```

Enable Quota Enforcement

> For BeeGFS versions older than 2015.03-r19
> setup in /etc/beegfs/beegfs-storage.conf
> (quotaEnableEnforcement = true)
> http://www.beegfs.com/wiki/EnableQuota#enableenforcement

- Update beegfs-mgmtd configuration file

```
# vi /etc/beegfs/beegfs-mgmtd.conf
...
quotaQueryGIDFile                       =
quotaQueryGIDRange                      = 1000,2000
quotaQueryUIDFile                       =
quotaQueryUIDRange                      = 1000,2000
quotaQueryType                          = system
quotaQueryWithSystemUsersGroups         = false
quotaUpdateIntervalMin                  = 10
...
quotaEnableEnforcement                  = true
...
```

- Restart beegfs-mgmtd

```
# systemctl restart beegfs-mgmtd
```

Set/Get Quota Limits

- Create user and set quota limits per user

> adduser
> beegfs-ctl --setquota --help
> beegfs-ctl --getquota --help

```
# adduser pdossant

# id pdossant
uid=1000(pdossant) gid=1000(pdossant) groupes=1000(pdossant)

# beegfs-ctl --setquota --sizelimit=10M --inodelimit=100 --uid pdossant
Set or updated 1 user quota limits on node: localhost.localdomain [ID: 1]

# beegfs-ctl --getquota --uid pdossant
      user/group     ||           size          ||    chunk files
     name     |  id  ||    used    |    hard    ||  used   |  hard
--------------|------||------------|------------||---------|---------
      pdossant|  1000||      0 Byte|   10.00 MiB||        0|      100
```

- Set quota to multiple users from a file

Each line of the file is made up of three fields
- ID/name
- size limit
- inode limit

```
# cat quota_limits.txt
pdossant,5M,0

# beegfs-ctl --setquota --uid --file=quota_limits.txt
Set or updated 1 user quota limits on node: localhost.localdomain [ID: 1]

# beegfs-ctl --getquota --uid pdossant
      user/group     ||           size          ||    chunk files
     name     |  id  ||    used    |    hard    ||  used   |  hard
--------------|------||------------|------------||---------|---------
      pdossant|  1000||      0 Byte|    5.00 MiB||        0|unlimited
```

- Test quota

```
# mkdir /mnt/beegfs/pdossant
# chown pdossant.pdossant /mnt/beegfs/pdossant

# su - pdossant

$ cd /mnt/beegfs/pdossant/

$ dd if=/dev/zero of=justafile bs=1k count=10000
10000+0 records in
10000+0 records out
10240000 bytes (10 MB) copied, 0.0460161 s, 223 MB/s

$ dd if=/dev/zero of=justafile1 bs=1k count=100000
dd: error writing ‘justafile1’: Disk quota exceeded
dd: closing output file ‘justafile1’: Disk quota exceeded

$ beegfs-ctl --getquota --uid pdossant
      user/group     ||           size          ||    chunk files
     name     |  id  ||    used    |    hard    ||  used   |  hard
--------------|------||------------|------------||---------|---------
      pdossant|  1000||  195.31 MiB|    5.00 MiB||        2|unlimited
```

What’s wrong?

Quota is checked every 10 minutes …

```
# vi /etc/beegfs/beegfs-mgmtd.conf
...
quotaUpdateIntervalMin                = 10
...
```

Statistics

Server

beegfs-ctl --serverstats --help

- Get metadata server stats

```
# beegfs-ctl --serverstats --nodetype=meta --perserver --interval=1
...
```

- Get storage server stats

```
# beegfs-ctl --serverstats --nodetype=storage --perserver --interval=1
...
```

Client

beegfs-ctl --clientstats --help

- Get metadata client stats

```
# beegfs-ctl --clientstats --nodetype=storage --names --interval=1
...
```

- Get storage client stats

```
# beegfs-ctl --clientstats --nodetype=meta --names --interval=1
...
```

User

beegfs-ctl --userstats --help

- Get metadata user stats

```
# beegfs-ctl --userstats  --nodetype=meta --names --interval=1
...
```

- Get storage user stats

```
# beegfs-ctl --userstats  --nodetype=meta --names --interval=1
...
```

File system check and repair

beegfs-fsck --help

While file system is in use

```
# beegfs-fsck --checkfs

BeeGFS File System Check Version : 6.2

----------------------------------------------------------------------
Started BeeGFS fsck in forward check mode [Wed Dec 14 10:36:40 2016]
Log will be written to /var/log/beegfs-fsck.log
Database will be saved in /var/lib/beegfs/
----------------------------------------------------------------------

Step 1: Check reachability of nodes: Finished

-----
Waiting for metadata servers to start modification logging. This might take some time.
-----

Step 2: Gather data from nodes:

Fetched data > Directory entries: 2 | Inodes: 4 | Chunks: 3


-----
Waiting for metadata servers to stop modification logging. This might take some time.
-----

Step 3: Check for errors...

* Duplicated inode IDs ...
* Duplicated chunks ...
* Duplicated content directores ...
* Bad target mirror information in dentry ...
* Bad content mirror information in dir inode ...
* Bad content mirror information in file inode ...
* Malformed chunk ...
* File has a missing target in stripe pattern ...
* Dentry-by-ID file is present, but no corresponding dentry ...
* Dentry-by-ID file is broken or missing ...
* Chunk without an inode pointing to it (orphaned chunk) ...
* Chunk is saved in wrong path ...
* Wrong owner node saved in inode ...
* Dentry points to inode on wrong node ...
* Content directory without an inode ...
* Dir inode without a dentry pointing to it (orphaned inode) ...
* File inode without a dentry pointing to it (orphaned inode) ...
* Dangling directory entry ...
* Directory inode without a content directory ...
* Attributes of file inode are wrong ...
* Attributes of dir inode are wrong ...
```

Skip repairs

```
# beegfs-fsck --checkfs --readonly


BeeGFS File System Check Version : 6.2

--------------------------------------------------------------------
Started BeeGFS fsck in forward check mode [Wed Dec 14 10:39:30 2016]
Log will be written to /var/log/beegfs-fsck.log
Database will be saved in /var/lib/beegfs/
--------------------------------------------------------------------

Step 1: Check reachability of nodes: Finished
The database path already exists: /var/lib/beegfs/fsckdb
If you continue now any existing database files in that path will be deleted.
Do you want to continue? (Y/N)
Y

-----
Waiting for metadata servers to start modification logging. This might take some time.
-----

Step 2: Gather data from nodes:

Fetched data > Directory entries: 2 | Inodes: 4 | Chunks: 3


-----
Waiting for metadata servers to stop modification logging. This might take some time.
-----

Step 3: Check for errors...

* Duplicated inode IDs ...
* Duplicated chunks ...
* Duplicated content directores ...
* Bad target mirror information in dentry ...
* Bad content mirror information in dir inode ...
* Bad content mirror information in file inode ...
* Malformed chunk ...
* File has a missing target in stripe pattern ...
* Dentry-by-ID file is present, but no corresponding dentry ...
* Dentry-by-ID file is broken or missing ...
* Chunk without an inode pointing to it (orphaned chunk) ...
* Chunk is saved in wrong path ...
* Wrong owner node saved in inode ...
* Dentry points to inode on wrong node ...
* Content directory without an inode ...
* Dir inode without a dentry pointing to it (orphaned inode) ...
* File inode without a dentry pointing to it (orphaned inode) ...
* Dangling directory entry ...
* Directory inode without a content directory ...
* Attributes of file inode are wrong ...
* Attributes of dir inode are wrong ...

Found 0 errors. Detailed information can also be found in /var/log/beegfs-fsck.out.
```

Repair errors

```
# beegfs-fsck --checkfs --automatic


BeeGFS File System Check Version : 6.2


--------------------------------------------------------------------
Started BeeGFS fsck in forward check mode [Wed Dec 14 10:42:16 2016]
Log will be written to /var/log/beegfs-fsck.log
Database will be saved in /var/lib/beegfs/
--------------------------------------------------------------------

Step 1: Check reachability of nodes: Finished
The database path already exists: /var/lib/beegfs/fsckdb
If you continue now any existing database files in that path will be deleted.
Do you want to continue? (Y/N)
y

-----
Waiting for metadata servers to start modification logging. This might take some time.
-----

Step 2: Gather data from nodes:

Fetched data > Directory entries: 2 | Inodes: 4 | Chunks: 3


-----
Waiting for metadata servers to stop modification logging. This might take some time.
-----

Step 3: Check for errors...

* Duplicated inode IDs ...
* Duplicated chunks ...
* Duplicated content directores ...
* Bad target mirror information in dentry ...
* Bad content mirror information in dir inode ...
* Bad content mirror information in file inode ...
* Malformed chunk ...
* File has a missing target in stripe pattern ...
* Dentry-by-ID file is present, but no corresponding dentry ...
* Dentry-by-ID file is broken or missing ...
* Chunk without an inode pointing to it (orphaned chunk) ...
* Chunk is saved in wrong path ...
* Wrong owner node saved in inode ...
* Dentry points to inode on wrong node ...
* Content directory without an inode ...
* Dir inode without a dentry pointing to it (orphaned inode) ...
* File inode without a dentry pointing to it (orphaned inode) ...
* Dangling directory entry ...
* Directory inode without a content directory ...
* Attributes of file inode are wrong ...
* Attributes of dir inode are wrong ...
```