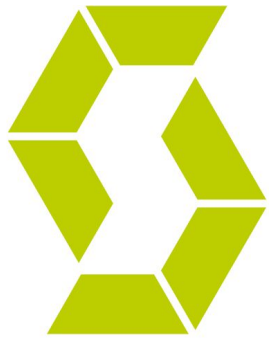

Getting Started Guide

StorPool version 16.01.1xx
Document version 2016-06-07

Introduction

StorPool is a distributed storage software. It pools the attached storage devices (hard disks or SSDs) to create a single shared storage pool and then provides standard block devices from that pool.

There are several daemons and kernel modules on a StorPool cluster node that provide different services:

- **Beacon**
  The **storpool_beacon** daemon must be the first StorPool process to be started on all nodes in the cluster. It takes care of informing all members about the availability of the node on which it is installed. Each beacon service makes sure that its node is seen by the others and counted in the quorum. It requires the **storpool_rdma** kernel module.

- **Server**
  The **storpool_server** daemon must be started on each node that provides hard drives and SSDs to the cluster. The service requires the **storpool_disk** kernel module.

- **Block**
  The **storpool_block** daemon provides the client functionality. StorPool volumes may only be attached to the nodes running this service. Requires the **storpool_bd** kernel module.

- **Management**
  The **storpool_mgmt** daemon receives requests from user-space tools (CLI or API), executes them in the StorPool cluster and returns the results back to the sender. The built-in automatic failover mechanism ensures high availability for the management service.

CLI Overview

StorPool provides an easy yet powerful Command Line Interface (CLI) for administration of the cluster. There are various ways to execute commands; which one will be used depends on the style and the needs of the administrator.

Type a regular shell command with parameters

```
# storpool service list
```

Use the interactive StorPool shell

```
# storpool
StorPool> service list
```

Pipe some command's output to the StorPool CLI

```
# echo "service list" | storpool
```

Redirect the standard input from a predefined file with commands

```
# storpool < input_file
```

The CLI has an integrated help system that provides useful information at every step.

Display the available command-line options

```
# storpool --help
```

An error message with the possible options will be displayed if the shell command is incomplete or wrong

```
# storpool service
Error: incomplete command! Expected:
       list - list Services

# storpool attach disk
Error: incomplete command! Expected:
       diskId (0..4095)
       list - list disks
```

The interactive shell help can be invoked by pressing the **?** key

```
# storpool
StorPool> service ?
  list - list Services
```

The shell autocompletion, invoked by two presses of the **Tab** key, will show the options available for the current step

```
# storpool
StorPool> volume <tab><tab>
<volumeName>  list          status
```

The StorPool shell can detect incomplete lines and suggest options

```
# storpool
StorPool> volume <enter>
..................^
Error: incomplete command! Expected:
       volumeName - VolumeName (string(200))
       status - status of volumes
       list - list Volumes
```

Cluster Details

The StorPool configuration file (/etc/storpool.conf) uses the .ini format. All variables that are not in a named section are common for all the hosts (e.g. SP_EXPECTED_NODES, SP_AUTH_TOKEN).
Node-specific values are placed in a section named after the hostname of the node.
For example, use this syntax to set ID 1 for the node with the hostname spserver1

```
[spserver1]
SP_OURID=1
```

Use the following command to check the current status of StorPool cluster, listing all services:

```
# storpool service list
cluster running, mgmt on node 1
```

```
  mgmt   1 running on node  1 ver 16.01.1, started 2016-05-12 12:38:27, uptime 18:26:24 active
  mgmt   2 running on node  2 ver 16.01.1, started 2016-05-12 12:39:48, uptime 18:25:03
server   1 running on node  1 ver 16.01.1, started 2016-05-12 20:12:54, uptime 10:51:57
server   2 running on node  2 ver 16.01.1, started 2016-05-12 20:18:30, uptime 10:46:21
server   3 running on node  3 ver 16.01.1, started 2016-05-12 20:12:59, uptime 10:51:52
client  10 running on node 10 ver 16.01.1, started 2016-05-13 06:10:08, uptime 00:54:43
client  11 running on node 11 ver 16.01.1, started 2016-05-13 06:26:09, uptime 00:38:42
```

The output above shows that in the cluster there are five nodes with IDs 1, 2, 3, 10 and 11. The nodes with IDs 1, 2, and 3 work as servers and provide their storage devices to the cluster. The nodes 10 and 11 have client functionality which means that StorPool volumes can be attached to them. The management service is enabled on nodes 1 and 2, and is currently active on the first node.

To see the cluster space usage, use:

```
# storpool disk list
disk | server |   size |   used | est.free |    % | free entries | on-disk | allocated objects
 101 |      1 | 193 GB | 136 GB |    49 GB | 73 % |       785342 |  123 GB |     6639 / 210000
 111 |      1 | 932 GB | 464 GB |   426 GB | 51 % |       862240 |  400 GB |    39516 / 975000
 121 |      1 | 932 GB | 469 GB |   421 GB | 52 % |       879256 |  407 GB |    39014 / 975000
 122 |      1 | 932 GB | 471 GB |   419 GB | 52 % |       878879 |  407 GB |    39014 / 975000
 201 |      2 | 224 GB | 145 GB |    69 GB | 67 % |       896687 |  127 GB |     6837 / 240000
 211 |      2 | 931 GB | 478 GB |   412 GB | 53 % |       857226 |  423 GB |    41418 / 975000
 212 |      2 | 466 GB | 286 GB |   160 GB | 63 % |       430911 |  289 GB |    29325 / 495000
 213 |      2 | 932 GB | 462 GB |   428 GB | 51 % |       856349 |  412 GB |    41041 / 975000
 301 |      3 | 194 GB | 138 GB |    48 GB | 74 % |       787812 |  127 GB |     6838 / 210000
 302 |      3 | 224 GB | 148 GB |    66 GB | 68 % |       886314 |  131 GB |     7038 / 240000
 321 |      3 | 932 GB | 476 GB |   414 GB | 53 % |       876001 |  417 GB |    40766 / 975000
 322 |      3 | 932 GB | 482 GB |   409 GB | 53 % |       876104 |  416 GB |    39670 / 975000
 323 |      3 | 932 GB | 482 GB |   409 GB | 53 % |       865860 |  424 GB |    40082 / 975000
-----------------------------------------------------------------------------------------------
  13 |      3 | 8.5 TB | 4.5 TB |   3.6 TB | 55 % |     10738981 |  4.0 TB |  377198 / 9195000
```

To see more detailed information about the disks:

```
# storpool disk list info
disk | server |   device  |            model |               serial | description | flags |
 101 |      1 | /dev/sdb1 | INTEL_SSDSC2BB24 |    BTWL34500AAA000NGA |           - |   ssd |
 111 |      1 | /dev/sdd1 | Hitachi_HUA72201 |        JPA0J0N13AAAAA |           - |       |
 121 |      1 | /dev/sde1 | WDC_WD1003FBYZ-0 |       WD-WCAW3A00AAAA |           - |       |
 122 |      1 | /dev/sda1 | WDC_WD1003FBYZ-0 |       WD-WCAW3A00AAAB |           - |       |
 201 |      2 | /dev/sdi1 | INTEL_SSDSC2BB24 |    BTWL34500AAA000NGB |           - |   ssd |
 211 |      2 | /dev/sdc1 |   9690SA-4I_DISK |  N13N630A0A000A00A000 |           - |       |
 212 |      2 | /dev/sdf1 | Hitachi_HDS72105 |        JP0000FR00AAAA |           - |       |
 213 |      2 | /dev/sdd1 | WDC_WD1003FBYZ-0 |       WD-WCAW3A00AAAC |           - |       |
 301 |      3 | /dev/sdd1 | INTEL_SSDSC2BB24 |    BTWL34500AAA000NGC |           - |   ssd |
 302 |      3 | /dev/sde1 | INTEL_SSDSC2BB24 |    BTWL34500AAA000NGD |           - |   ssd |
 321 |      3 | /dev/sdf1 | WDC_WD1003FBYZ-0 |       WD-WCAW3A00AAAD |           - |       |
 322 |      3 | /dev/sda1 | WDC_WD1003FBYZ-0 |       WD-WCAW3A00AAAE |           - |       |
 323 |      3 | /dev/sdb1 | WDC_WD1003FBYZ-0 |       WD-WCAW3A00AAAF |           - |       |
```

The upper two tables are also helpful in determining which disk ID is assigned to the physical storage device and which server it is attached to.

Volumes

Volumes are the basic service of the StorPool storage system. They have a name and a certain size. They can be read from and written to. They can be attached to hosts as read-only or read-write block devices under /dev/storpool.

Before creating a volume, one or more placement groups must exist in the cluster. These groups are predefined sets of disks, over which the volume's objects will be replicated. It is possible to specify which individual disks to add to the group or alternatively to add all the disks provided by the specified server. In the following examples two groups will be created.

The first contains all the SSDs

```
# storpool placementGroup ssd addDisk 101 addDisk 201 addDisk 301 addDisk 302
OK
# storpool placementGroup ssd list
 type   | id
 disk   | 101 201 301 302
```

The second contains all the HDDs

```
# storpool placementGroup hdd addDisk 111 addDisk 121 addDisk 122
OK
# storpool placementGroup hdd addDisk 211 addDisk 212 addDisk 213
OK
# storpool placementGroup hdd addDisk 321 addDisk 322 addDisk 323
OK
# storpool placementGroup hdd list
 type   | id
 disk   | 111 121 122 211 212 213 321 322 323
```

Now create a volume with a given name, size, and replication, using the already created placement groups

```
# storpool volume testvol1 size 1G replication 3 placeAll hdd placeTail ssd
OK
# storpool volume list
-------------------------------------------------------------------------------
|   volume | size | repl. | placeAll | placeTail | iops | bw | parent | template |
-------------------------------------------------------------------------------
| testvol1 | 1 GB |    3  |      hdd |       ssd |    - |  - |      - |          |
-------------------------------------------------------------------------------
```

When a large number of similar volumes must be created, the administrator can predefine attributes in a template and then base volumes on it.

Define a template

```
# storpool template hybrid-r3 size 1G replication 3 placeAll hdd placeTail ssd
OK
```

Now create a volume based on this template

```
# storpool volume testvol2 template hybrid-r3
```

```
OK
# storpool volume list
-------------------------------------------------------------------------------------
|   volume | size | repl | placeAll | placeTail | iops | bw | parent |  template |
-------------------------------------------------------------------------------------
| testvol1 | 1 GB |    3 |      hdd |       ssd |    - |  - |      - |           |
| testvol2 | 1 GB |    3 |      hdd |       ssd |    - |  - |      - | hybrid-r3 |
-------------------------------------------------------------------------------------
```

Snapshots
Snapshots are read-only point-in-time images of volumes. They are created once and cannot be changed. They can be attached to hosts as read-only block devices.

Creating a snapshot of a volume.

```
# storpool volume testvol1 snapshot ss_testvol1
OK
# storpool snapshot list
-----------------------------------------------------------------------------------------------------------
|    snapshot | size | repl | placeAll | placeTail |          created on |   volume | iops | bw | parent | template |
-----------------------------------------------------------------------------------------------------------
| ss_testvol1 | 1 GB |    3 |      hdd |       ssd | 2015-03-01 14:25:00 | testvol1 |    - |
 - |        |          |
-----------------------------------------------------------------------------------------------------------
```

Volumes can be based on snapshots. Such volumes contain only the changes since the snapshot was taken. After a volume is created from a snapshot, any data written will be recorded within the volume. Read requests may be served by the volume (for data that has been written to it) or by its parent snapshot.

To create a volume based on an existing snapshot

```
# storpool volume testvol3 parent ss_testvol1
OK
# storpool volume list
-------------------------------------------------------------------------------------
|   volume | size | repl | placeAll | placeTail | iops | bw |      parent |  template |
-------------------------------------------------------------------------------------
| testvol1 | 1 GB |    3 |      hdd |       ssd |    - |  - | ss_testvol1 |           |
| testvol2 | 1 GB |    3 |      hdd |       ssd |    - |  - |           - | hybrid-r3 |
| testvol3 | 1 GB |    3 |      hdd |       ssd |    - |  - | ss_testvol1 |           |
-------------------------------------------------------------------------------------
```

Use Volumes and Snapshots

Attaching a volume or snapshot makes it accessible to the client as a block device in the /dev/storpool directory. Volumes can be attached as read-only or read-write. Snapshots are attached as read-only.

Attach a volume to the client node with ID 10; this creates the /dev/storpool/testvol1 block device

```
# storpool attach volume testvol1 client 10
```

Attach a snapshot to client 20

```
# storpool attach volume ss_testvol1 client 20
```

List all the attachments

```
# storpool attach list
-------------------------------------------------------------
| client | volume                                     | mode |
-------------------------------------------------------------
|     10 | testvol1                                   | RW   |
|     20 | ss_testvol1                                | RO   |
-------------------------------------------------------------
```

On a node where a volume is attached, it can be used as a regular block device

```
[root@node10]# ls -l /dev/storpool/testvol1
lrwxrwxrwx 1 root root 7 Apr 21 20:00 /dev/storpool/testvol1 -> ../sp-0

[root@node10]# fdisk -l /dev/storpool/testvol1

Disk /dev/storpool/testvol1: 1073 MB, 1073741824 bytes
64 heads, 32 sectors/track, 1024 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

Testing Volumes
After a volume has been created and attached to a client, it is possible to perform synthetic performance tests over it using the **fio** tool. *Please note that the following examples can destroy all data on the tested volumes*.

If the volume is unallocated, the **storpool_server** will return zeroes from memory, without actually reading any data from the drives. So to get realistic performance results, first it will be necessary to fill the volume end-to-end.

```
[root@node10]# fio --name=fill --ioengine=aio --direct=1 --sync=1 --rw=write --bs=1M
--iodepth=1 --filename=/dev/storpool/testvol1
```

To measure the latency times of random operations on the volume, use **iodepth=1**

```
[root@node10]# fio --name=latency-randwrite --ioengine=aio --direct=1 --randrepeat=0
--sync=1 --rw=randwrite --bs=4k --iodepth=1 --filename=/dev/storpool/testvol1

[root@node10]# fio --name=latency-randread --ioengine=aio --direct=1 --randrepeat=0
--sync=1 --rw=randread --bs=4k --iodepth=1--filename=/dev/storpool/testvol1
```

To measure the simulated real-life performance in input/output operations per second (iops), use random operations, set the blocksize to a value which will be used by the filesystem (e.g. 4KB) and set **iodepth** to a value approximating the number of concurrent read/write requests that an actual application would use (e.g. higher for a webserver handling many simultaneous requests or for a database engine handling queries from/to different databases or different tables)

```
[root@node10]# fio --name=iops-randwrite --ioengine=aio --direct=1 --randrepeat=0
--rw=randwrite --bs=4k --iodepth=128 --filename=/dev/storpool/testvol1

[root@node10]# fio --name=iops-randread --ioengine=aio --direct=1 --randrepeat=0
--rw=randread --bs=4k --iodepth=128 --filename=/dev/storpool/testvol1
```

Use sequential operations with a big block size (e.g. 1MB) to measure the throughput of the volume

```
[root@node10]# fio --name=sequential-writes --ioengine=aio --direct=1 --randrepeat=0
--rw=write --bs=1m --iodepth=128 --filename=/dev/storpool/testvol1

[root@node10]# fio --name=sequential-reads --ioengine=aio --direct=1 --randrepeat=0
--rw=read --bs=1m --iodepth=128 --filename=/dev/storpool/testvol1
```