MooseFS 3.0 Storage Classes Manual

Core Technology Development & Support Team

June 8, 2016

Proofread by Agata Kruszona-Zawadzka
Coordination & layout by Piotr Robert Konopelko.

Please send corrections to Piotr Robert Konopelko – peter@mfs.io.

Contents

Chapter 1

Introduction to Storage Classes functionality in MooseFS 3.0

1.1 What is a Storage Class?

Since MooseFS 3.0 goal has been extended to Storage Class. Storage Classes allow you to specify on which Chunkservers copies of files should be stored. Storage Classes are defined using label expressions.

To maintain compatibility with standard goal semantics, there are predefined Storage Classes from 1 to 9 that, unless changed (see **Section 3.1.7: Predefined Storage Classes** of this manual or `man mfsscadmin`), behave like goals from MooseFS 2.0 or 1.6. Goal tools simply work only on these classes.

1.2 What are labels?

Labels are letters (A-Z – 26 letters) that can be assigned to Chunkservers. Each chunkserver can have multiple (up to 26) labels.

Labels expression is a set of subexpressions separated by commas, each subexpression specifies the storage schema of one copy of a file. Subexpression can be: an asterisk or a label schema.

Label schema can be one label or an expression with sums, multiplications and brackets. Sum means a file can be stored on any chunkserver matching any element of the sum (logical or). Multiplication means a file can be stored only on a chunkserver matching all elements (logical and). Asterisk means any chunkserver.

Identical subexpressions can be shortened by adding a number in front of one instead of repeating it a number of times.

For more information about labels expressions, refer to **Section 3.1.5: Labels expressions** of this manual.

Chapter 2

How to use Storage Classes?

2.1 Machines configuration

In this example we have MooseFS 3.0 installed on 11 machines:

- `ts02`, `ts03` – Master Servers
- `ts04..ts12` – Chunkservers

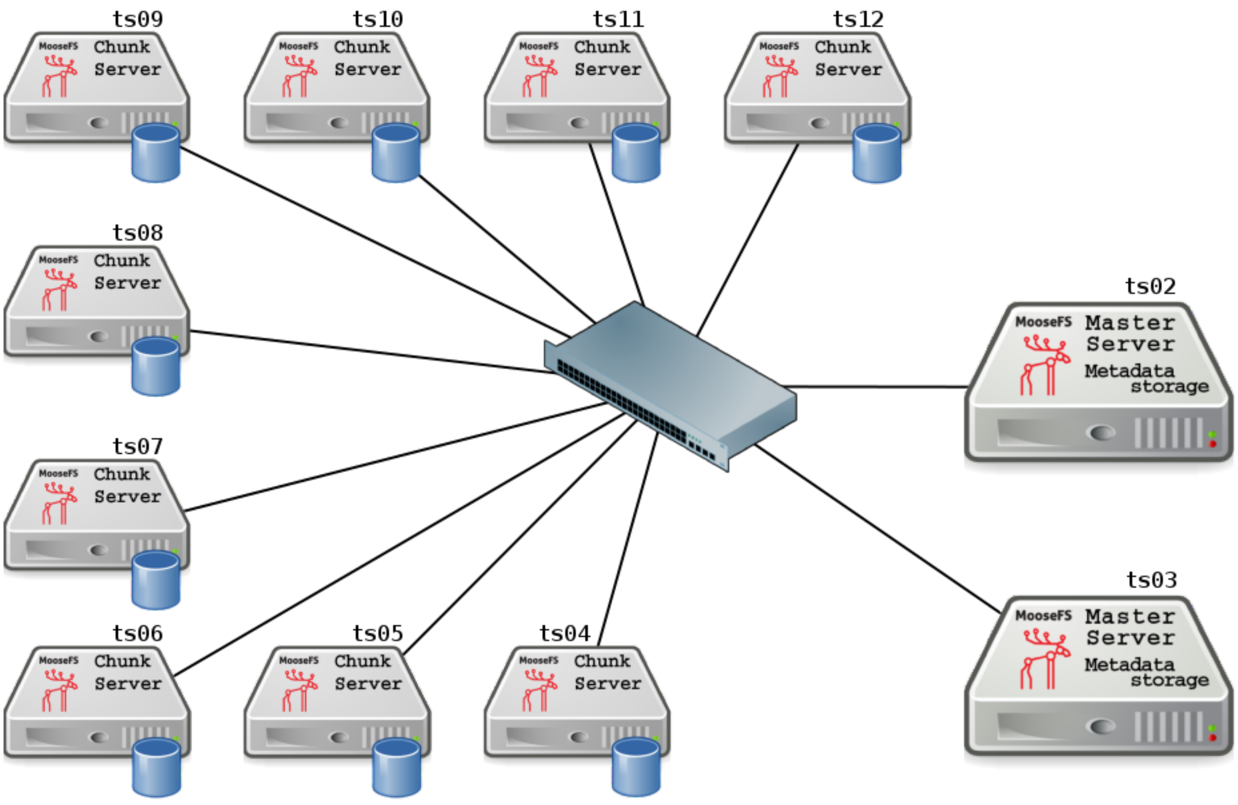
Assumption:

- On the MooseFS instance there is some initial data stored with goal `2` (Storage Class `2`).

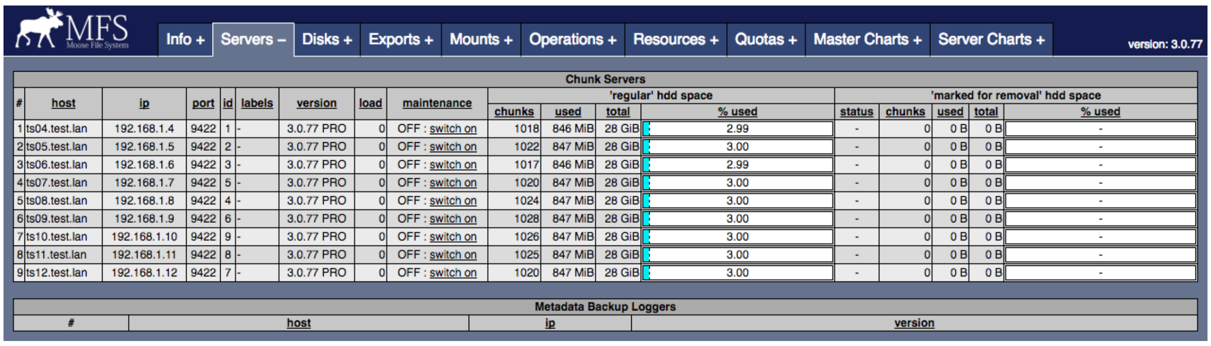
2.2 Example of MooseFS installation without Storage Classes

To run MooseFS without any user-defined Storage Classes, you don't have to make any changes in configuration. Just install MooseFS with default configuration. The process is described in "**MooseFS Step by Step Tutorial**".

The picture below shows the discussed installation:



If labels on Chunkservers are not set up, the system is balanced like MooseFS 2.0. The image below presents system balance at this point:


MFS Moose File System
Info + | Servers – | Disks + | Exports + | Mounts + | Operations + | Resources + | Quotas + | Master Charts + | Server Charts +
version: 3.0.77

Chunk Servers

#	host	ip	port	id	labels	version	load	maintenance	'regular' hdd space				'marked for removal' hdd space				
									chunks	used	total	% used	status	chunks	used	total	% used
1	ts04.test.lan	192.168.1.4	9422	1	-	3.0.77 PRO	0	OFF : switch on	1018	846 MiB	28 GiB	2.99	-	0	0 B	0 B	-
2	ts05.test.lan	192.168.1.5	9422	2	-	3.0.77 PRO	0	OFF : switch on	1022	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
3	ts06.test.lan	192.168.1.6	9422	3	-	3.0.77 PRO	0	OFF : switch on	1017	846 MiB	28 GiB	2.99	-	0	0 B	0 B	-
4	ts07.test.lan	192.168.1.7	9422	5	-	3.0.77 PRO	0	OFF : switch on	1020	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
5	ts08.test.lan	192.168.1.8	9422	4	-	3.0.77 PRO	0	OFF : switch on	1024	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
6	ts09.test.lan	192.168.1.9	9422	6	-	3.0.77 PRO	0	OFF : switch on	1028	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
7	ts10.test.lan	192.168.1.10	9422	9	-	3.0.77 PRO	0	OFF : switch on	1026	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
8	ts11.test.lan	192.168.1.11	9422	8	-	3.0.77 PRO	0	OFF : switch on	1025	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
9	ts12.test.lan	192.168.1.12	9422	7	-	3.0.77 PRO	0	OFF : switch on	1020	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-

Metadata Backup Loggers

#	host	ip	version

2.3 Labelling Chunkservers

To add labels to the system, i.e. assign them to Chunkservers, you need to edit their configuration files (`/etc/mfs/mfschunkserver.cfg`). Open the file, uncomment the following line and after the equation character type labels you want to set on specific Chunkserver. For example to set label `A` on Chunkservers ts04, ts05, ts06 and ts07, their configuration should look like this:

```
[...]

# labels string (default is empty - no labels)
LABELS = A

[...]
```

The next step is to "inform" the Chunkserver, that the Configuration file has changed. Issue the command:

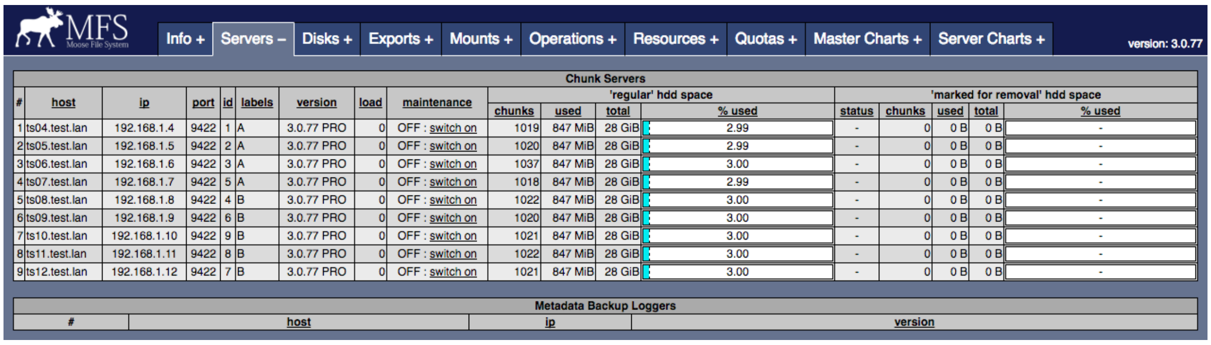
```
root@chunkserver:~# service moosefs-pro-chunkserver reload
```

or:

```
root@chunkserver:~# mfschunkserver reload
```

Similarly set label `B` for Chunkservers ts08, ts09, ts10, ts11, ts12.

After this step in CGI monitor you can observe, that Chunkservers ts04..ts07 have label `A` and Chunkservers ts08..ts12 – label `B`:



Chunk Servers

#	host	ip	port	id	labels	version	load	maintenance	'regular' hdd space				'marked for removal' hdd space				
									chunks	used	total	% used	status	chunks	used	total	% used
1	ts04.test.lan	192.168.1.4	9422	1	A	3.0.77 PRO	0	OFF : switch on	1019	847 MiB	28 GiB	2.99	-	0	0 B	0 B	-
2	ts05.test.lan	192.168.1.5	9422	2	A	3.0.77 PRO	0	OFF : switch on	1020	847 MiB	28 GiB	2.99	-	0	0 B	0 B	-
3	ts06.test.lan	192.168.1.6	9422	3	A	3.0.77 PRO	0	OFF : switch on	1037	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
4	ts07.test.lan	192.168.1.7	9422	5	A	3.0.77 PRO	0	OFF : switch on	1018	847 MiB	28 GiB	2.99	-	0	0 B	0 B	-
5	ts08.test.lan	192.168.1.8	9422	4	B	3.0.77 PRO	0	OFF : switch on	1022	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
6	ts09.test.lan	192.168.1.9	9422	6	B	3.0.77 PRO	0	OFF : switch on	1020	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
7	ts10.test.lan	192.168.1.10	9422	9	B	3.0.77 PRO	0	OFF : switch on	1021	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
8	ts11.test.lan	192.168.1.11	9422	8	B	3.0.77 PRO	0	OFF : switch on	1022	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-
9	ts12.test.lan	192.168.1.12	9422	7	B	3.0.77 PRO	0	OFF : switch on	1021	847 MiB	28 GiB	3.00	-	0	0 B	0 B	-

Metadata Backup Loggers

#	host	ip	version

Notice: If you want to set more than one label for a Chunkserver, just enter appropriate labels in configuration file (`/etc/mfs/mfschunkserver.cfg`). MooseFS supports schemes listed below, so you can choose the one, which fits for you the best, e.g.:

```
[...]
# labels string (default is empty - no labels)
LABELS = XYZ
[...]
```

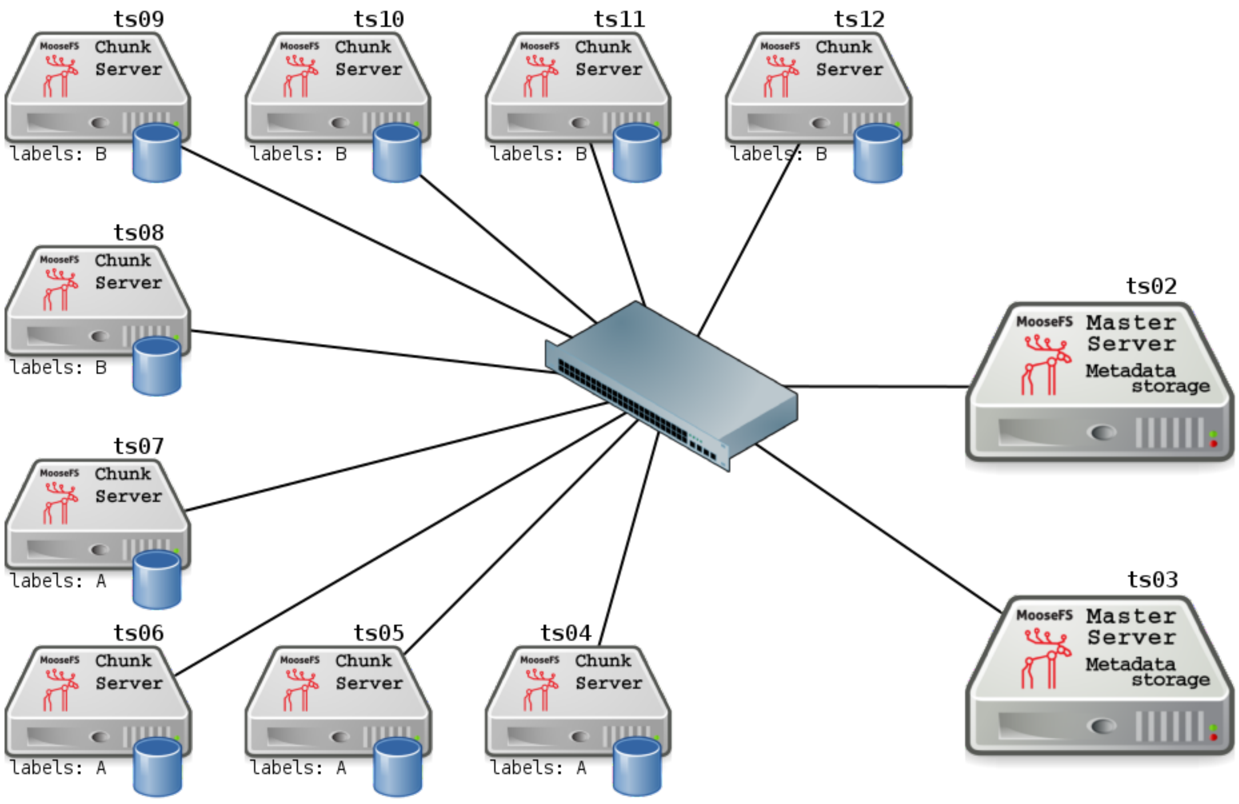
or:

```
[...]
# labels string (default is empty - no labels)
LABELS = X, Y, Z
[...]
```

or:

```
[...]
# labels string (default is empty - no labels)
LABELS = X Y Z
[...]
```

The picture below presents current system configuration:



2.4 Creating Storage Classes

In order to create a Storage Class on MooseFS, use the `mfsscadmin` tool. Below you can find a simple example, you can read a full description of `mfsscadmin` usage in **Chapter 3: Storage Classes tools** or in `man mfsscadmin`.

Let's create a storage class named `sclass1`:

First of all, mount MooseFS:

Listing 2.1: Mounting MooseFS (Linux only)

```
root@client:~# mount -t moosefs mfsmaster.test.lan: /mnt/mfs
mfsmaster 192.168.1.2 - found leader: 192.168.1.3
mfsmaster accepted connection with parameters: read-write,restricted_ip,admin ;
    root mapped to root:root
root@client:~#
```

or

Listing 2.2: Mounting MooseFS (universal)

```
root@client:~# mfsmount -H mfsmaster.test.lan /mnt/mfs
mfsmaster 192.168.1.2 - found leader: 192.168.1.3
mfsmaster accepted connection with parameters: read-write,restricted_ip,admin ;
    root mapped to root:root
root@client:~#
```

Then, navigate to mounted file system:

```
root@client:~# cd /mnt/mfs
root@client:/mnt/mfs#
```

Let's assume, you want to have your files stored in 2 copies on Chunkservers labelled as `A`. Create a Storage Class with appropriate definition:

```
root@client:/mnt/mfs# mfsscadmin create 2A sclass1
create ; 0
storage class make sclass1: ok
root@client:/mnt/mfs#
```

It means that every file with `sclass1` assigned will be stored in two copies: one will be kept on Chunkserver with label A, another one – on another Chunkserver with label A.

Similarly, create a Storage Class `sclass2`, which keep 2 copies on Chunkservers labelled as `B`:

```
root@client:/mnt/mfs# mfsscadmin create 2B sclass2
create ; 0
storage class make sclass2: ok
root@client:/mnt/mfs#
```

Notice: You don't have to navigate to mounted file system to create a Storage Class – it is also possible to do it from any location. In such case just let `mfsscadmin` tool know, where MooseFS is mounted (in first parameter), e.g.:

```
root@client:~# mfsscadmin /mnt/mfs create 2B sclass2
```

It applies to all Storage Classes tools.

2.5 Listing Storage Classes

Now, let's check, if the classes has been properly created and are available to use:

```
root@client:/mnt/mfs# mfsscadmin list
list ; 1
1
2
3
4
5
6
7
8
9
sclass1
sclass2
root@client:/mnt/mfs#
```

You can also see more detailed view by issuing the command with `-l` switch:

```
root@client:/mnt/mfs# mfsscadmin list -l
list ; 1
[...]
sclass1 : 2 ; admin_only: NO ; create_mode: STD ; create_labels: [A] , [A] ;
    keep_labels: [A] , [A]
sclass2 : 2 ; admin_only: NO ; create_mode: STD ; create_labels: [B] , [B] ;
    keep_labels: [B] , [B]
root@client:/mnt/mfs#
```

2.6 Assigning Storage Class to files / directories

There are several tools to manage Storage Classes assignment to files, directories etc.: `mfsgetsclass`, `mfssetsclass`, `mfscopysclass`, `mfsxchgsclass`, `mfslistsclass`. You can find out more about them in **Section 3.2: MooseFS Storage Class management tools – `mfssclass`** or by issuing `man mfssclass`.

Now it's time to store some data on this MooseFS instance. Create two directories, let's say `dataX` and `dataY`.

```
root@client:~# cd /mnt/mfs
root@client:/mnt/mfs# mkdir dataX
root@client:/mnt/mfs# mkdir dataY
root@client:/mnt/mfs#
```

Next, assign Storage class `sclass1` to `/mnt/mfs/dataX`:

```
root@client:/mnt/mfs# mfssetsclass sclass1 dataX
dataX: storage class: 'sclass1'
root@client:/mnt/mfs#
```

It means that this directory, its subdirectories, files and so on will be stored according to `sclass1` policy.

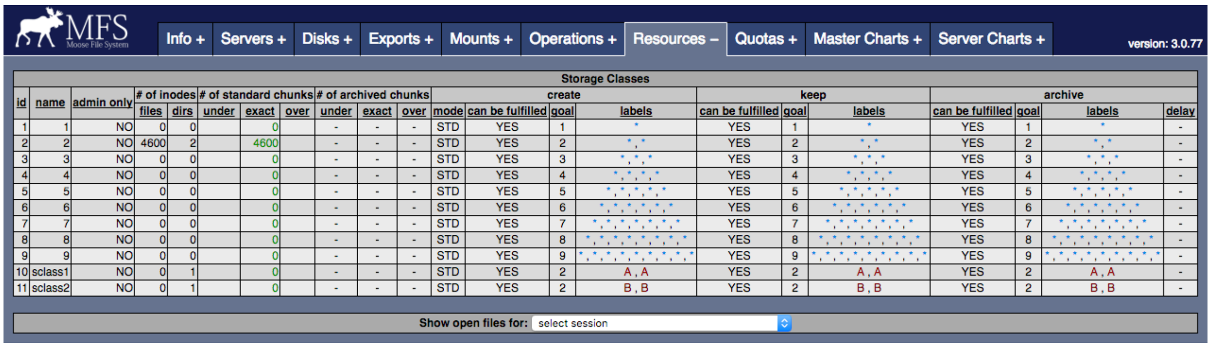
Similarly, assign Storage class `sclass2` to `/mnt/mfs/dataY`:

```
root@client:/mnt/mfs# mfssetsclass sclass2 dataY
dataY: storage class: 'sclass2'
root@client:/mnt/mfs#
```

It means that this directory, its subdirectories, files and so on will be stored according to `sclass2` policy.

For more information about assigning Storage Classes to files, refer to **Section 3.2: MooseFS Storage Class management tools – `mfssclass`**.

Now on MooseFS Monitor ("Resources" tab) you can observe, that goal is set and it can be fulfilled.



Storage Classes

			# of inodes		# of standard chunks			# of archived chunks			create				keep			archive			
id	name	admin only	files	dirs	under	exact	over	under	exact	over	mode	can be fulfilled	goal	labels	can be fulfilled	goal	labels	can be fulfilled	goal	labels	delay
1	1	NO	0	0		0		-	-	-	STD	YES	1	*	YES	1	*	YES	1	*	-
2	2	NO	4600	2		4600		-	-	-	STD	YES	2	*, *	YES	2	*, *	YES	2	*, *	-
3	3	NO	0	0		0		-	-	-	STD	YES	3	*, *, *	YES	3	*, *, *	YES	3	*, *, *	-
4	4	NO	0	0		0		-	-	-	STD	YES	4	*, *, *, *	YES	4	*, *, *, *	YES	4	*, *, *, *	-
5	5	NO	0	0		0		-	-	-	STD	YES	5	*, *, *, *, *	YES	5	*, *, *, *, *	YES	5	*, *, *, *, *	-
6	6	NO	0	0		0		-	-	-	STD	YES	6	*, *, *, *, *, *	YES	6	*, *, *, *, *, *	YES	6	*, *, *, *, *, *	-
7	7	NO	0	0		0		-	-	-	STD	YES	7	*, *, *, *, *, *, *	YES	7	*, *, *, *, *, *, *	YES	7	*, *, *, *, *, *, *	-
8	8	NO	0	0		0		-	-	-	STD	YES	8	*, *, *, *, *, *, *, *	YES	8	*, *, *, *, *, *, *, *	YES	8	*, *, *, *, *, *, *, *	-
9	9	NO	0	0		0		-	-	-	STD	YES	9	*, *, *, *, *, *, *, *, *	YES	9	*, *, *, *, *, *, *, *, *	YES	9	*, *, *, *, *, *, *, *, *	-
10	sclass1	NO	0	1		0		-	-	-	STD	YES	2	A , A	YES	2	A , A	YES	2	A , A	-
11	sclass2	NO	0	1		0		-	-	-	STD	YES	2	B , B	YES	2	B , B	YES	2	B , B	-

Show open files for: select session

2.6.1 Creating files

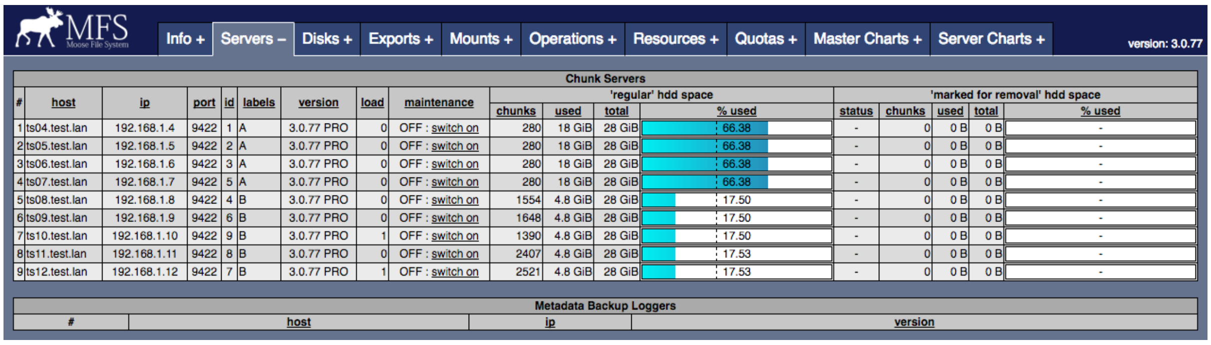
In this step you will create some files in previously created directories (`labelA` and `labelB`) to fill MooseFS instance with data. This operation may take some time. Issue the following commands:

```
root@client:/mnt/mfs# cd dataX
root@client:/mnt/mfs/dataX# for i in `seq 1 35`; do dd if=/dev/urandom of=
    dd1G_$i.bin bs=1M count=1024; done
[...]
root@client:/mnt/mfs/dataX# cd ../dataY
root@client:/mnt/mfs/dataY# for i in `seq 1 10`; do dd if=/dev/urandom of=
    dd1G_$i.bin bs=1M count=1024; done
[...]
root@client:/mnt/mfs/dataY#
```

Notice: These commands create approx. 90 GiB (45 GiB multiplied by goal 2) of data – 35 GiB in `dataX` directory (RAW size: 70 GiB) and 10 GiB in `dataY` directory (RAW size: 20 GiB), so adjust them for your testing purposes.

2.6.2 Filesystem balance with Storage Classes applied

Now you can observe, that filesystem is balanced according to Storage Classes policy: Chunkservers with label A store the data data with goal `2A` applied, similarly – Chunkservers with label B store the data with goal `2B`:



Chunk Servers

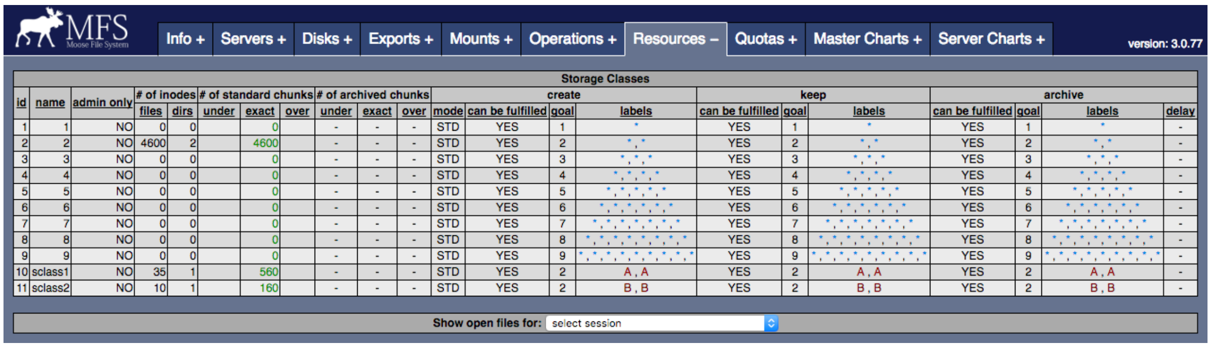
#	host	ip	port	id	labels	version	load	maintenance	'regular' hdd space: chunks	used	total	% used	'marked for removal' hdd space: status	chunks	used	total	% used
1	ts04.test.lan	192.168.1.4	9422	1	A	3.0.77 PRO	0	OFF : switch on	280	18 GiB	28 GiB	66.38	-	0	0 B	0 B	-
2	ts05.test.lan	192.168.1.5	9422	2	A	3.0.77 PRO	0	OFF : switch on	280	18 GiB	28 GiB	66.38	-	0	0 B	0 B	-
3	ts06.test.lan	192.168.1.6	9422	3	A	3.0.77 PRO	0	OFF : switch on	280	18 GiB	28 GiB	66.38	-	0	0 B	0 B	-
4	ts07.test.lan	192.168.1.7	9422	5	A	3.0.77 PRO	0	OFF : switch on	280	18 GiB	28 GiB	66.38	-	0	0 B	0 B	-
5	ts08.test.lan	192.168.1.8	9422	4	B	3.0.77 PRO	0	OFF : switch on	1554	4.8 GiB	28 GiB	17.50	-	0	0 B	0 B	-
6	ts09.test.lan	192.168.1.9	9422	6	B	3.0.77 PRO	0	OFF : switch on	1648	4.8 GiB	28 GiB	17.50	-	0	0 B	0 B	-
7	ts10.test.lan	192.168.1.10	9422	9	B	3.0.77 PRO	1	OFF : switch on	1390	4.8 GiB	28 GiB	17.50	-	0	0 B	0 B	-
8	ts11.test.lan	192.168.1.11	9422	8	B	3.0.77 PRO	0	OFF : switch on	2407	4.8 GiB	28 GiB	17.53	-	0	0 B	0 B	-
9	ts12.test.lan	192.168.1.12	9422	7	B	3.0.77 PRO	1	OFF : switch on	2521	4.8 GiB	28 GiB	17.53	-	0	0 B	0 B	-

Metadata Backup Loggers

#	host	ip	version

Notice, that the system looks "unbalanced", but it is, in fact, balanced as much, as the requirements of Storage Classes allow it to be.

Also in tab "Resources" number of inodes has changed:



Storage Classes

id	name	admin only	# of inodes: files	dirs	# of standard chunks: under	exact	over	# of archived chunks: under	exact	over	create: mode	can be fulfilled	goal	labels	keep: can be fulfilled	goal	labels	archive: can be fulfilled	goal	labels	delay
1	1	NO	0	0		0		-	-	-	STD	YES	1	*	YES	1	*	YES	1	*	-
2	2	NO	4600	2		4600		-	-	-	STD	YES	2	*, *	YES	2	*, *	YES	2	*, *	-
3	3	NO	0	0		0		-	-	-	STD	YES	3	*, *, *	YES	3	*, *, *	YES	3	*, *, *	-
4	4	NO	0	0		0		-	-	-	STD	YES	4	*, *, *, *	YES	4	*, *, *, *	YES	4	*, *, *, *	-
5	5	NO	0	0		0		-	-	-	STD	YES	5	*, *, *, *, *	YES	5	*, *, *, *, *	YES	5	*, *, *, *, *	-
6	6	NO	0	0		0		-	-	-	STD	YES	6	*, *, *, *, *, *	YES	6	*, *, *, *, *, *	YES	6	*, *, *, *, *, *	-
7	7	NO	0	0		0		-	-	-	STD	YES	7	*, *, *, *, *, *, *	YES	7	*, *, *, *, *, *, *	YES	7	*, *, *, *, *, *, *	-
8	8	NO	0	0		0		-	-	-	STD	YES	8	*, *, *, *, *, *, *, *	YES	8	*, *, *, *, *, *, *, *	YES	8	*, *, *, *, *, *, *, *	-
9	9	NO	0	0		0		-	-	-	STD	YES	9	*, *, *, *, *, *, *, *, *	YES	9	*, *, *, *, *, *, *, *, *	YES	9	*, *, *, *, *, *, *, *, *	-
10	sclass1	NO	35	1		560		-	-	-	STD	YES	2	A , A	YES	2	A , A	YES	2	A , A	-
11	sclass2	NO	10	1		160		-	-	-	STD	YES	2	B , B	YES	2	B , B	YES	2	B , B	-

Show open files for: select session

2.7 Creation, keep, archive labels

In MooseFS 3.0 a possibility to "plan" changing labels has been added.

Now you can "tell" MooseFS (crate appropriate Storage Class), what label expression it should use for file(s) while creating it (them), to what label expression change it after the creation and to what label expression change it after a specific time since last modification.

You can define it while creating a Storage Class by `mfsscadmin` tool.

2.7.1 Synopsis

```
mfsscadmin [/MOUNTPOINT] create|make [-a admin_only] [-m creation_mode]
[-C CREATION_LABELS] -K KEEP_LABELS [-A ARCH_LABELS -d ARCH_DELAY] SCLASS_NAME...
```

2.7.2 Creation labels

"Creation labels" (`-C CREATION_LABELS`) – optional parameter, that tells the system to which Chunkservers, defined by the `CREATION_LABELS` expression, the chunk should be first written just after creation; if this parameter is not provided for a class, the `KEEP_LABELS` Chunkservers will be used.

2.7.3 Keep labels

"Keep labels" (`-K KEEP_LABELS`) – mandatory parameter (assumed in the second, abbreviated version of the command), that tells the system on which Chunkservers, defined by the `KEEP_LABELS` expression, the chunk(s) should be kept always, except for special conditions like creating and archiving, if defined.

2.7.4 Archive labels

"Archive labels" (`-A ARCH_LABELS -d ARCH_DELAY`) – optional parameter, that tells the system on which Chunkservers, defined by the `ARCH_LABELS` expression, the chunk(s) should be kept for archiving purposes; the system starts to treat a chunk as archive, when the last modification time (`mtime`) of the file it belongs to is older than the number of days specified with `-d` parameter.

2.7.5 How to set it?

For more information about the command to issue, refer to **Section 3.1: MooseFS Storage Class administration tool** – `mfsscadmin` or issue `man mfsscadmin`.

2.8 Chunkserver states

Chunkserver can work in 3 states: `normal`, `overloaded` and (since MooseFS 3.0.62) `internal rebalance`:

- `Normal` state is a standard state. In "Servers" CGI tab you can see load as a normal number, e.g.: `7`.
- `Internal rebalance` state is a special Chunkserver state. It is activated when e.g. you add a new, empty HDD to a Chunkserver. Then Chunkserver enters this special mode and rebalances chunks between all HDDs to make all HDDs utilization as close to equal as possible. In "Servers" CGI tab you can see load as number in round brackets, e.g.: `(7)`.
- `Overloaded` is a special, **temporary** Chunkserver state. It is activated when Chunkserver load is high and Chunkserver is not able to perform more operations at the moment. In such case, Chunkserver sends an information to Master Server that it is overloaded. If the load lowers to the normal level, Chunkserver sends an information to Master Server, that it is not overloaded any more. In "Servers" CGI tab you can see load as a number in square brackets, e.g.: `[77]`.

2.9 Chunk creation modes

While you store your data on labelled Chunkservers, a situation may occur that there is no more space on appropriate Chunkservers or they are overloaded.

To decide what MooseFS should do when free space ends or when Chunkserver you want to store data to is overloaded, you need to use creating chunks modes.

You can define these modes for each file, directory, it's subdirectories and so on, because they can be set (or modified) when you set the goal for your data.

There are three modes:

- `loose` mode (`-m L` flag to `mfsscadmin`) – in this mode the system will use other servers in case of overloaded servers or no space on servers and will replicate data to correct servers when it becomes possible.
- `default` mode (no flag or `-m D` flag to `mfsscadmin`) – in case of overloaded servers system will wait for them, but in case of no space available will use other servers and will replicate data to correct servers when it becomes possible.
- `strict` mode (`-m S` flag to `mfsscadmin`) – in this mode the system will return error (`ENOSPC`) in case of no space available on servers marked with labels specified for chunk creation. It will still wait for overloaded servers.

A table below presents MooseFS behavior for these modes:

	Chunkserver is full	**Chunkserver is overloaded**
Loose	use servers with other labels	use servers with other labels
Default	use servers with other labels	wait for available Chunkserver
Strict	no write (returns `ENOSPC`)	wait for available Chunkserver

You can observe current states in Resources CGI tab.

2.10 Preferred labels during read/write (in `mfsmount`)

It is possible to specify preferred labels for choosing Chunkservers during read and write operations at the MooseFS Client (`mfsmount`) side:

```
-o mfspreflabels=LABELEXPR
     specify preferred labels for choosing Chunkservers during I/O
```

You can set different preferred labels for each mountpoint.

Preferred labels in MooseFS Client are a list (up to 9) of labels expressions, e.g. E_1, E_2, E_3.

While a client performs a read operation, Master Server returns a list of chunks' locations (in random order) in the following form (CS means Chunkserver): CS_a, CS_b, CS_c, ...

Each of CS_x entry contains a list of labels assigned to specific Chunkserver.

Priority of each CS_x is calculated as the minimum y value, where labels from CS_x match expression E_y. If no expression matches, the priority is set as a number of expressions $+1$.

The lowest number means the highest priority.

Then, the list of Chunkservers is sorted by priorities. The first Chunkserver from the list (which has the highest priority / the lowest number) is used while reading.

If more than one Chunkserver has the same priority, Client picks the one that got the least number of operations from this Client so far.

If a specific chunk read ends with an error, Client can use a chunk copy with lower priority (greater number).

In case of writing, the list of Chunkservers is sorted similarly and data is written to Chunkserver with the highest priority. The difference is, if more that one Chunkserver has the same priority, the order form Master Server is used.

If no `mfspreflabels` is set, the order of list from MooseFS Master is used with no further modifications.

Chapter 3

Storage Classes tools

3.1 MooseFS Storage Class administration tool – `mfsscadmin`

3.1.1 Synopsis

- `mfsscadmin [/MOUNTPOINT] create|make [-a admin_only] [-m creation_mode] [-C CREATION_LABELS] -K KEEP_LABELS [-A ARCH_LABELS -d ARCH_DELAY] SCLASS_NAME...`
- `mfsscadmin [/MOUNTPOINT] create|make [-a admin_only] [-m creation_mode] LABELS SCLASS_NAME...`
- `mfsscadmin [/MOUNTPOINT] change|modify [-f] [-a admin_only] [-m creation_mode] [-C CREATION_LABELS] [-K KEEP_LABELS] [-A ARCH_LABELS] [-d ARCH_DELAY] SCLASS_NAME...`
- `mfsscadmin [/MOUNTPOINT] delete|remove SCLASS_NAME...`
- `mfsscadmin [/MOUNTPOINT] copy|duplicate SRC_SCLASS_NAME DST_SCLASS_NAME...`
- `mfsscadmin [/MOUNTPOINT] rename SRC_SCLASS_NAME DST_SCLASS_NAME`
- `mfsscadmin [/MOUNTPOINT] list [-l]`

3.1.2 Description

`mfsscadmin` is a tool for defining storage classes, which can be later applied to MooseFS objects with mfssetsclass, mfsgetsclass etc.

Storage class is a set of labels expressions and options that indicate, on which chunkservers the files in this class should be written and later kept.

3.1.3 Commands

- `create|make` creates a new storage class with given options, described below and names it `SCLASS_NAME`; there can be more than one name provided, multiple storage classes with the same definition will be created then

- `change|modify` – changes the given options in a class or classes indicated by `SCLASS_NAME` paremeter(s)
- `delete|remove` – removes the class or classes indicated by `SCLASS_NAME` paremeter(s); if any of the classes is not empty (i.e. it is still used by some MooseFS objects), it will not be removed and the tool will return an error and an error message will be printed; empty classes will be removed in any case
- `copy|duplicate` – copies class indicated by `SRC_SCLASS_NAME` under a new name provided with `DST_SCLASS_NAME`
- `rename` – changes the name of a class from `SRC_SCLASS_NAME` to `DST_SCLASS_NAME`
- `list` – lists all the classes

3.1.4 Options

- `-C` – optional parameter, that tells the system to which chunkservers, defined by the `CREATION_LABELS` expression, the chunk should be first written just after creation; if this parameter is not provided for a class, the `KEEP_LABELS` chunkservers will be used
- `-K` – mandatory parameter (assumed in the second, abbreviated version of the command), that tells the system on which chunkservers, defined by the `KEEP_LABELS` expression, the chunk(s) should be kept always, except for special conditions like creating and archiving, if defined
- `-A` – optional parameter, that tells the system on which chunkservers, defined by the `ARCH_LABELS` expression, the chunk(s) should be kept for archiving purposes; the system starts to treat a chunk as archive, when the last modification time of the file it belongs to is older than the number of days specified with `-d` option
- `-d` – optional parameter that **must** be defined when `-A` is defined, `ARCH_DELAY` parameter defines after how many days from last modification time a file (and its chunks) are treated as archive
- `-a` – can be either 1 or 0 and indicates if the storage class is available to everyone (0) or admin only (1)
- `-f` – force the changes on a predefined storage class (see below), use with caution!
- `-m` – is described below in "Creation modes" section
- `-l` – list also definitions, not only the names of existing storage classes

3.1.5 Labels expressions

Labels are letters (A-Z – 26 letters) that can be assigned to chunkservers. Each chunkserver can have multiple (up to 26) labels. Labels are defined in `mfschunkserver.cfg` file, for more information refer to the appropriate manpage.

Labels expression is a set of subexpressions separated by commas, each subexpression specifies the storage schema of one copy of a file. Subexpression can be: an asterisk or a label schema.

Label schema can be one label or an expression with sums, multiplications and brackets. Sum means a file can be stored on any chunkserver matching any element of the sum (logical or).

Multiplication means a file can be stored only on a chunkserver matching all elements (logical and). Asterisk means any chunkserver. Identical subexpressions can be shortened by adding a number in front of one instead of repeating it a number of times.

Examples of labels expressions:

- `A,B` – files will have two copies, one copy will be stored on chunkserver(s) with label `A`, the other on chunkserver(s) with label `B`
- `A,*` – files will have two copies, one copy will be stored on chunkserver(s) with label `A`, the other on any chunkserver(s)
- `*,*` – files will have two copies, stored on any chunkservers (different for each copy)
- `AB,C+D` – files will have two copies, one copy will be stored on any chunkserver(s) that has both labels `A` and `B` (**multiplication of labels**), the other on any chunkserver(s) that has either the `C` label or the `D` label (**sum of labels**)
- `A,B[X+Y],C[X+Y]` – files will have three copies, one copy will be stored on any chunkserver(s) with A label, the second on any chunkserver(s) that has the `B` label and either `X` or `Y` label, the third on any chunkserver(s), that has the `C` label and either `X` or `Y` label
- `A,A` expression is equivalent to `2A` expression
- `A,BC,BC,BC` expression is equivalent to `A,3BC` expression
- `*,*` expression is equivalent to `2*` expression is equivalent to `2` expression

3.1.6 Creation modes

It is important to specify what to do in case when there is no space available on all servers marked with labels needed for new chunk creation. Also all servers marked with such labels can be temporarily overloaded. The question is if the system should create chunks on other servers or not.

Answer to this question should be resolved by user and hence the `-m` option.

- By default (no options or option `-m D`) in case of overloaded servers system will wait for them, but in case of no space available will use other servers and will replicate data to correct servers when it becomes possible.
- Option `-m S` turns on `STRICT` mode. In this mode the system will return error (`ENOSPC`) in case of no space available on servers marked with labels specified for chunk creation. It will still wait for overloaded servers.
- Option `-m L` turns on `LOOSE` mode. In this mode the system will use other servers in case of overloaded servers or no space on servers and will replicate data to correct servers when it becomes possible.

3.1.7 Predefined Storage Classes

For compatibility reasons, every fresh or freshly upgraded instance of MooseFS has 9 predefined storage classes. Their names are single digits, from `1` to `9`, and their definitions are `*` to `9*`.

They are equivalents of simple numeric goals from previous versions of the system. In case of an upgrade, all files that had goal `N` before upgrade, will now have `N` storage class.

These classes can be modified only when option `-f` is specified. It is advised to create new storage classes in an upgraded system and migrate files with mfsxchgsclass tool, rather than modify the predefined classes. The predefined classes **cannot** be deleted.

3.2 MooseFS Storage Class management tools – `mfssclass`

3.2.1 Synopsis

- `mfsgetsclass [-r] [-n|-h|-H|-k|-m|-g] OBJECT...`
- `mfssetsclass [-r] [-n|-h|-H|-k|-m|-g] SCLASS_NAME OBJECT...`
- `mfscopysclass [-r] [-n|-h|-H|-k|-m|-g] SOURCE_OBJECT OBJECT...`
- `mfsxchgsclass [-r] [-n|-h|-H|-k|-m|-g] SRC_SCLASS_NAME DST_SCLASS_NAME OBJECT...`
- `mfslistsclass [-l] [MOUNT_POINT]`

3.2.2 Description

These tools operate on object's Storage Class name. This is an extended version of classic goal. There are predefined storage classes provided as equivalents of goals 1 to 9 (names are simply `1`, `2`, ... , `9`). Other classes can be created / modified / deleted etc. by administrator using `mfsscadmin` tool.

- `mfsgetsclass` prints current storage class of given object(s). `-r` option enables recursive mode, which works as usual for every given file, but for every given directory additionally prints current storage class of all contained objects (files and directories).
- `mfssetsclass` changes current storage class of given object(s). `-r` option enables recursive mode.
- `mfscopysclass` copies storage class from one object to given object(s).
- `mfsxchgsclass` sets storage class to `DST_SCLASS_NAME` of given objects(s) but only when current storage class is set to `SRC_SCLASS_NAME`.
- `mfslistsclass` lists currently defined storage classes. `-l` option enables long format – whole class definition is printed for each class, not only its name. For description of storage class definition refer to mfsscadmin manpage.

3.2.3 General options

Most of mfstools use `-n`, `-h`, `-H`, `-k`, `-m` and `-g` options to select format of printed numbers.

- `-n` causes to print exact numbers,
- `-h` uses binary prefixes (Ki, Mi, Gi as 2^{10}, 2^{20} etc.) while `-H` uses SI prefixes (k, M, G as 10^3, 10^6 etc.).
- `-k`, `-m` and `-g` show plain numbers respectivaly in kibis (binary kilo – 1024), mebis (binary mega – 1024^2) and gibis (binary giga – 1024^3).

The same can be achieved by setting `MFSHRFORMAT` environment variable to: 0 (exact numbers), 1 or h (binary prefixes), 2 or H (SI prefixes), 3 or h+ (exact numbers and binary prefixes), 4 or H+ (exact numbers and SI prefixes). The default is to print just exact numbers.

3.2.4 Inheritance

When new object is created in MooseFS, attributes such as storage class, trashtime and extra attributes are inherited from parent directory. So if you set i.e. "noowner" attribute and storage class to "important" in a directory then every new object created in this directory will have storage class set to "important" and "noowner" flag set.

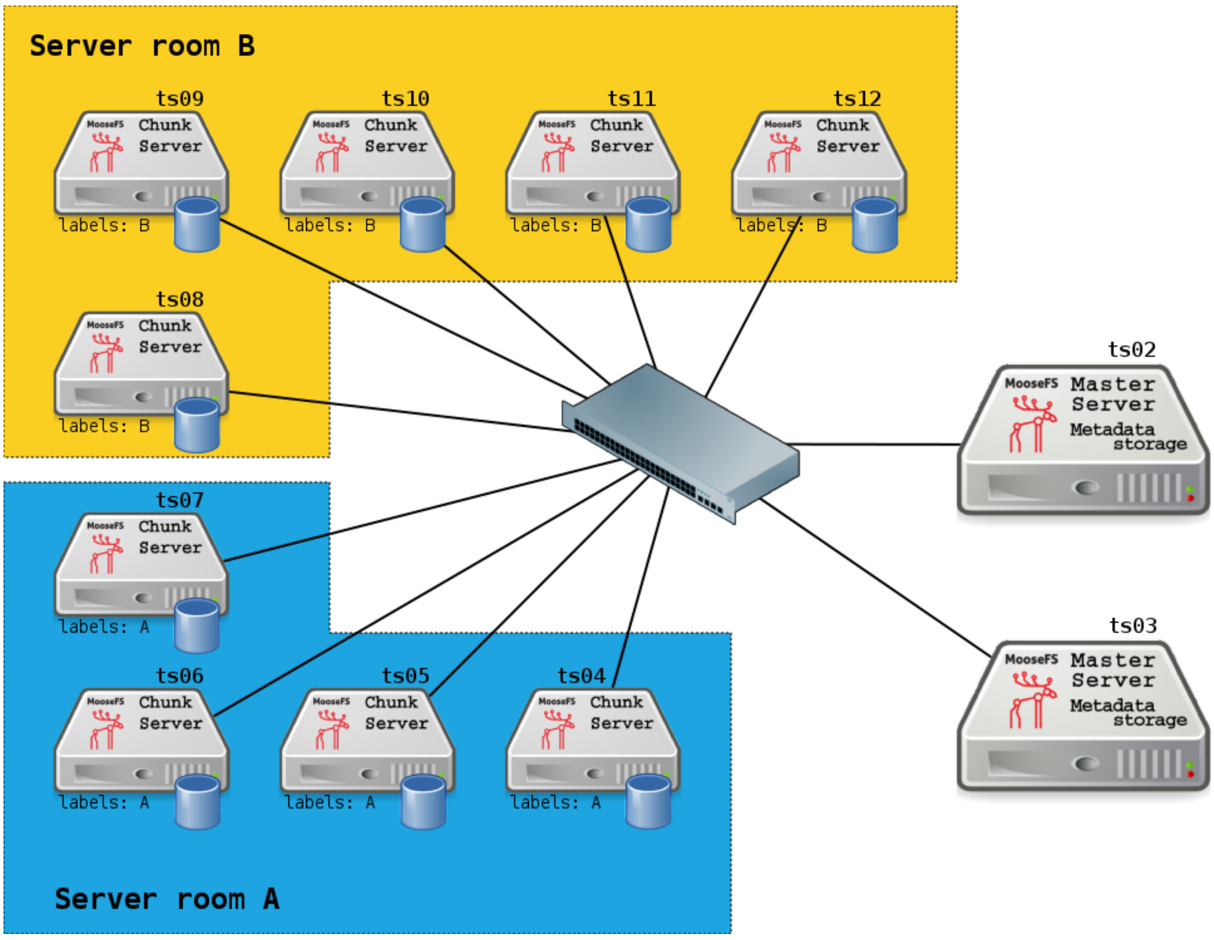
A newly created object inherits always the current set of its parent's attributes. Changing a directory attribute does not affect its already created children. To change an attribute for a directory and all of its children use `-r` option.

Chapter 4

Common use scenarios

4.1 Scenario 1: Two server rooms (A and B)

Let's assume that chunkservers with label A are in server room A, and with label B – in server room B (divided exactly as in steps above):

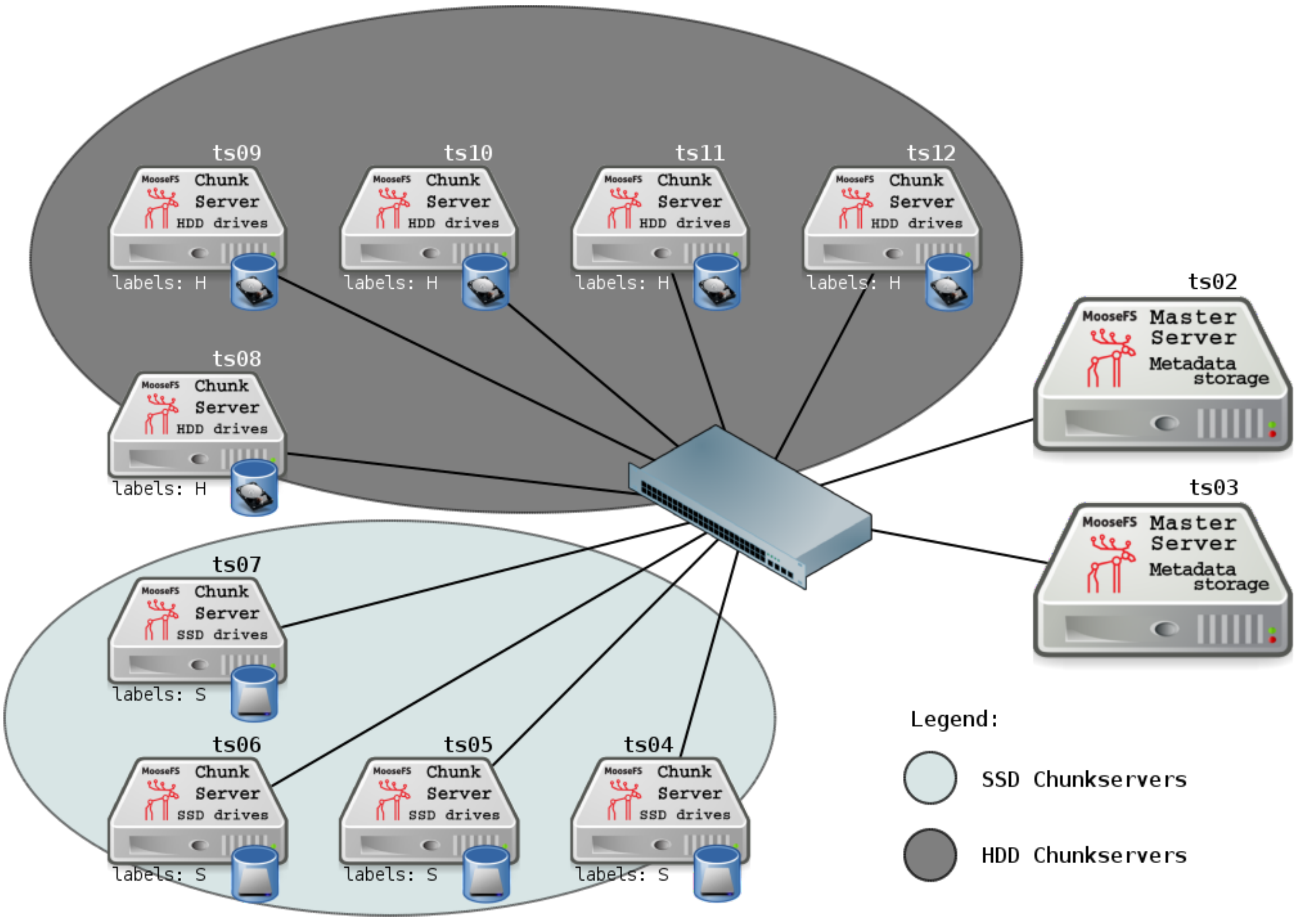


Using Storage Classes, you can simply decide, which server room your data is stored to.

Notice: Slow link between the sites (server room A and server room B in above example) will slow down I/O write operations to files with chunks stored in both sites due to synchronous nature of I/O write operations. Because of that reason alone, it is recommended to have a very fast connection between sites.

4.2 Scenario 2: SSD and HDD drives

Let's assume, that chunkservers ts04..ts07 have SSD drives and chunkservers ts08..ts12 have HDD drives. For example, you can label chunkservers with HDD drives as `H`, and with SSD drives – as `S`:



You can configure Storage Classes, so that your frequently used data is stored on SSD Chunkservers (e.g. Storage Class `ssd`), and data not accessed very often – on HDD Chunkservers (e.g. Storage Class `hdd`).

You can also easily move some data (e.g. after end of the year) from SSD to HDD chunkservers – you just need to change the Storage Class assignment from `ssd` to `hdd` for this data and MooseFS will automatically take care of moving process.

Example: you have a directory named `Reports2015` located on MooseFS mountpoint. This directory and its subdirectories and files are used very often by a lot of processes. You want to:

- store this directory in four copies – these are very important files

- speed up access to this directory,

so you set up and define a Storage Class e.g. `4ssdcopies` defined as `4S` (four copies on Chunkservers with fast, SSD drives) and assign it to the directory recursively. Issue the commands below:

```
root@client:~# cd /mnt/mfs

root@client:/mnt/mfs# mfsscadmin create 4S 4ssdcopies
create ; 0
storage class make 4ssdcopies: ok

root@client:/mnt/mfs# mfssetsclass -r 4ssdcopies Reports2015
Reports2015:
 inodes with storage class changed:              5685
 inodes with storage class not changed:          0
 inodes with permission denied:                  0

root@client:/mnt/mfs#
```

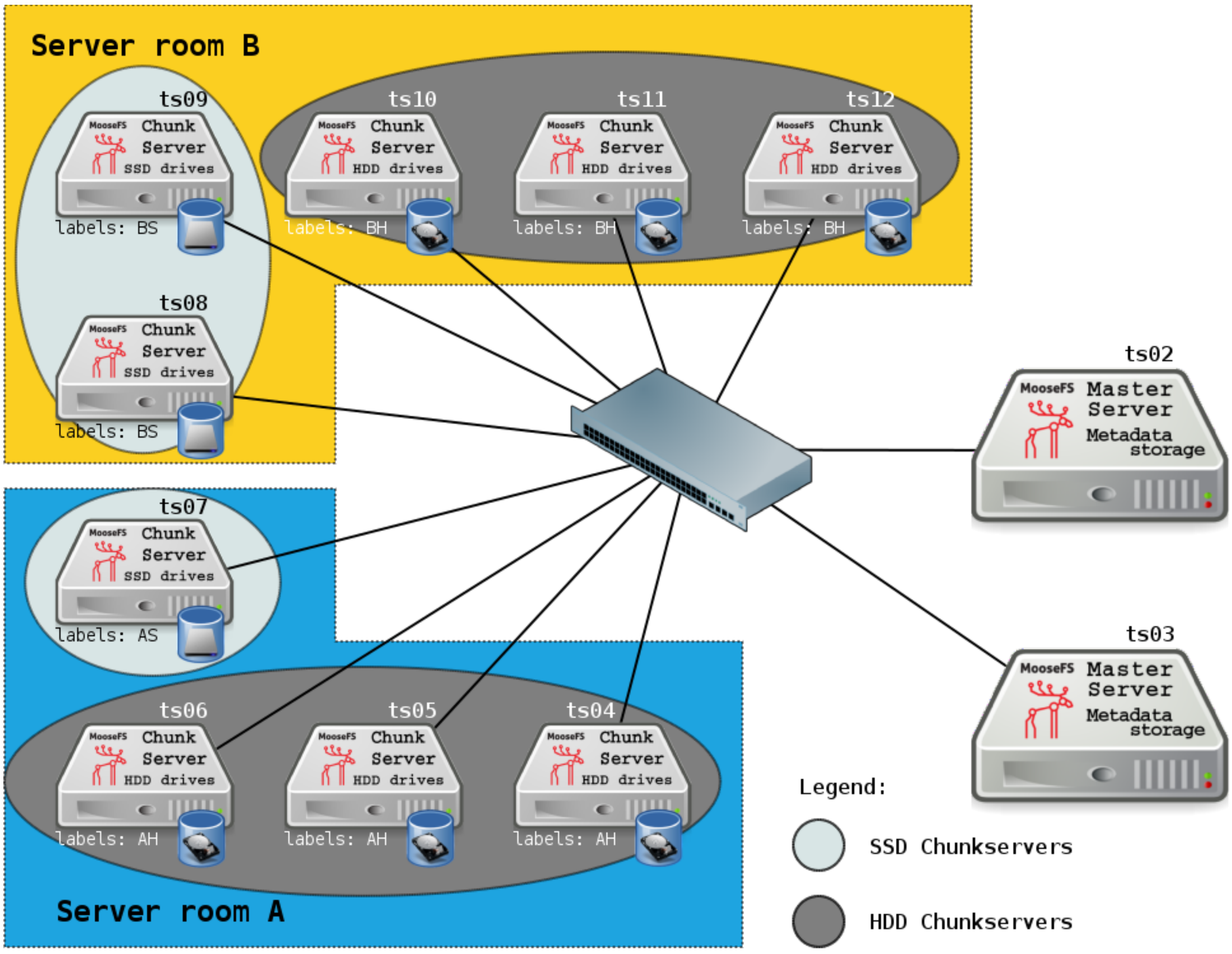
But year 2015 has passed, and now `Reports2015` is used infrequently and you want to free some space on SSD drives to store new data. So you want to move this directory, its subdirectories and files to HDD drives and store it only in three copies.

You just need to set up and define a Storage Class e.g. `3hddcopies` defined as `3H` (three copies on Chunkservers with HDD drives) and exchange the Storage Class for files which currently have `4ssdcopies` Storage Class applied with `3hddcopies` Storage Class:

```
root@client:~# cd /mnt/mfs

root@client:/mnt/mfs# mfsscadmin create 3H 3hddcopies
create ; 0
storage class make 3hddcopies: ok

root@client:/mnt/mfs# mfsxchgsclass -r 4ssdcopies 3hddcopies Reports2015
Reports2015:
 inodes with storage class changed:              5685
 inodes with storage class not changed:          0
 inodes with permission denied:                  0

root@client:/mnt/mfs#
```

MooseFS takes care of moving process and your data is safe and accessible during moving from SSD to HDD drives (Chunkservers).

4.3 Scenario 3: Two server rooms (A and B) + SSD and HDD drives


Server room B
ts09
MooseFS Chunk Server SSD drives
labels: BS
ts10
MooseFS Chunk Server HDD drives
labels: BH
ts11
MooseFS Chunk Server HDD drives
labels: BH
ts12
MooseFS Chunk Server HDD drives
labels: BH
ts08
MooseFS Chunk Server SSD drives
labels: BS
ts02
MooseFS Master Server Metadata storage
ts07
MooseFS Chunk Server SSD drives
labels: AS
ts03
MooseFS Master Server Metadata storage
ts06
MooseFS Chunk Server HDD drives
labels: AH
ts05
MooseFS Chunk Server HDD drives
labels: AH
ts04
MooseFS Chunk Server HDD drives
labels: AH
Server room A
Legend:
SSD Chunkservers
HDD Chunkservers

As shown in the picture above, this Scenario is a combination of Scenario 1 and Scenario 2. Let's assume, that in two server rooms you have two types of chunkservers: some of them containing HDD drives, some – SSD drives.

Now you want to store e.g. frequently used data on chunkservers with SSD drives and data used from time to time – on chunkservers with HDD drives. You also want to have a copy of all data in each server room.

In scenario presented above, you need to set the following labels:

- Server room A, SSD chunkservers: labels `A` and `S`,
- Server room A, HDD chunkservers: labels `A` and `H`,
- Server room B, SSD chunkservers: labels `B` and `S`,
- Server room B, HDD chunkservers: labels `B` and `H`.

Then you need to set up and define appropriate Storage Classes and apply them to your files.

Directory used very often named `Frequent` – you want to store it in 2 copies on **SSD** drives

(Chunkservers): one copy in server room `A`, another in server room `B`.

```
root@client:~# cd /mnt/mfs

root@client:/mnt/mfs# mfsscadmin create AS,BS frequent
create ; 0
storage class make frequent: ok

root@client:/mnt/mfs# mfssetsclass -r frequent Frequent
Frequent:
 inodes with storage class changed:              564513
 inodes with storage class not changed:          0
 inodes with permission denied:                  0

root@client:/mnt/mfs#
```

Directory used from time to time named `Rare` – you want to store it in 2 copies on **HDD** drives (Chunkservers): one copy in server room `A`, another in server room `B`.

```
root@client:~# cd /mnt/mfs

root@client:/mnt/mfs# mfsscadmin create AH,BH rare
create ; 0
storage class make rare: ok

root@client:/mnt/mfs# mfssetsclass -r rare Rare
Rare:
 inodes with storage class changed:              497251
 inodes with storage class not changed:          0
 inodes with permission denied:                  0

root@client:/mnt/mfs#
```

So your directory `Frequent` (and its subdirectories and files) is stored now on Chunkservers which have both `A` and `S` labels and on Chunkservers having both `B` and `S` labels.

Your directory `Rare` (and its subdirectories and files) is stored now on Chunkservers which have both `A` and `H` labels and on Chunkservers having both `B` and `H` labels.

You also want to store your directory named `Backup` in three copies. You want to store one copy in server room `A` on SSD chunkservers, and two copies in server room `B`, either on HDD or SSD chunkservers. Issue the following commands:

```
root@client:~# cd /mnt/mfs

root@client:/mnt/mfs# mfsscadmin create AS,2B[H+S] backup
create ; 0
storage class make backup: ok

root@client:/mnt/mfs# mfssetsclass -r backup Backup
Backup:
 inodes with storage class changed:              879784
 inodes with storage class not changed:          0
 inodes with permission denied:                  0

root@client:/mnt/mfs#
```

The labels expression `AS,2B[H+S]` is a *multiplication* and *sum* of labels. For more information, refer to **Section 3.1.5: Labels expressions** of this document.

For more information about `mfsscadmin` and `mfssetsclass`, refer to **Chapter 3: Storage Classes tools** of this document.

Notice: Slow link between the sites (server room A and server room B in above example) will slow down I/O write operations to files with chunks stored in both sites due to synchronous nature of I/O write operations. Because of that reason alone, it is recommended to have a very fast connection between sites.

4.4 Scenario 4: Creation, Keep and Archive modes

Let's assume you want to write fast a big amount of important data and your computer is located closer to server room A than to server room B. So you want to create chunks in server room A, on SSD chunkservers, in two copies (`-C 2AS`).

But your goal is to have one copy of this data in server room A, and the other one in server room B, both on SSD chunkservers. MooseFS will take care of the replication process (`-K AS,BS`).

And finally, after 30 days, you want MooseFS to move this data to HDD chunkservers in both server room A and B (`-A AH,BH -d 30`).

First of all, create a directory:

```
root@client:~# cd /mnt/mfs
root@client:/mnt/mfs# mkdir ImportantFiles
```

Then, set up and define a Storage Class, e.g. `important`, defined as `-C 2AS -K AS,BS -A AH,BH -d 30` and assign it to the newly created directory directory:

```
root@client:~# cd /mnt/mfs

root@client:/mnt/mfs# mfsscadmin create -C 2AS -K AS,BS -A AH,BH -d 30 important
create ; 0
storage class make important: ok

root@client:/mnt/mfs# mfssetsclass important ImportantFiles
ImportantFiles:
 inodes with storage class changed:              1
 inodes with storage class not changed:          0
 inodes with permission denied:                  0

root@client:/mnt/mfs#
```

And that's all! Now you can write the data to this directory.

Your data will be safe, stored very fast on SSD chunkservers in server room A while creating (you are close to this server room), copied by MooseFS also to server room B and after 30 days – automatically moved to HDD chunkservers.