NCFS Developer’s Guide

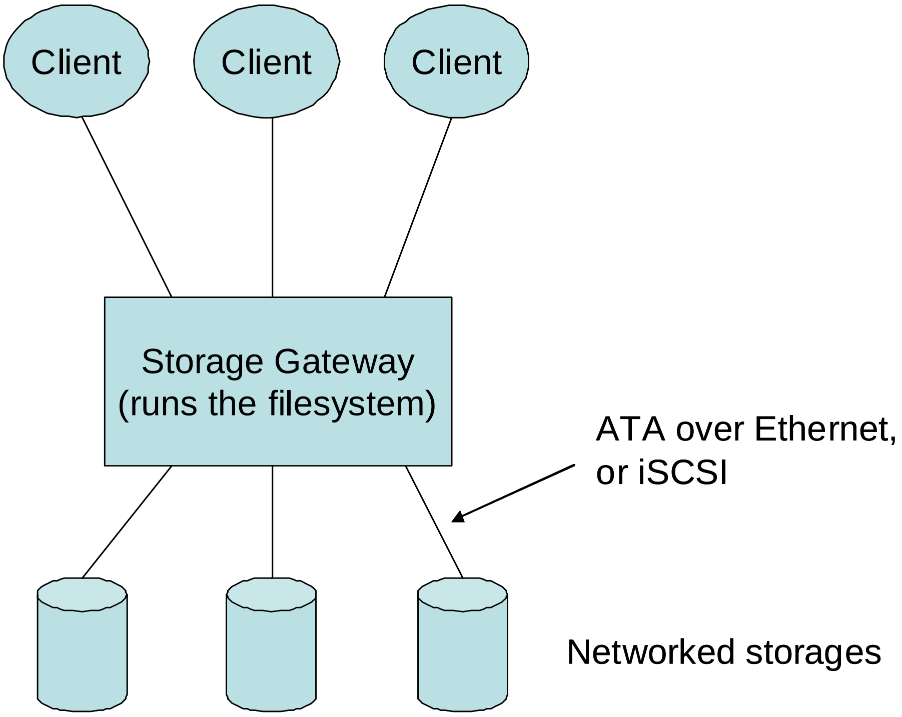
NCFS version: 1.02
Date: June 2011

Authors:
Lee Pak Ching (pclee@cse.cuhk.edu.hk)
Hu YuChong (yuchonghu@gmail.com)
Yu Chiu Man (cmyu@cse.cuhk.edu.hk)
Li Yan Kit (ykli7@cse.cuhk.edu.hk)
Kong Chun Ho (chkong8@cse.cuhk.edu.hk)
Zhu YunFeng (zyfl@mail.ustc.edu.cn)
Lui Chi Shing (cslui@cse.cuhk.edu.hk)

Contents:

I. INTRODUCTION

NCFS (Erasure Coding File System) is a distributed file system supporting erasure coding. A storage gateway running NCFS can connect with networked storage devices through ATA over Ethernet or iSCSI. The gateway can encode and decode the data on the storages according to customizable coding modules.



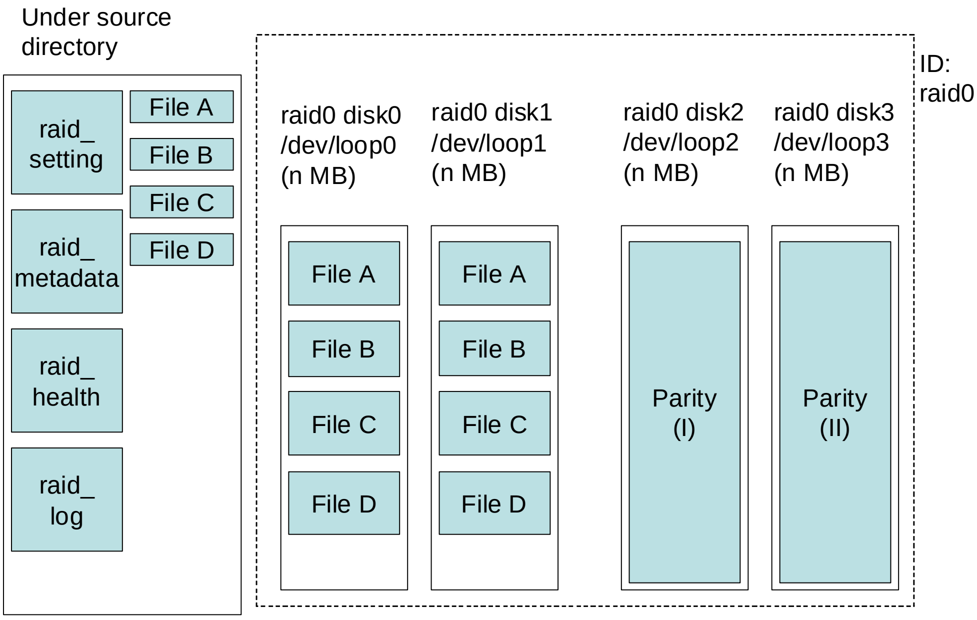
Disk layout:

NCFS reads setting files at the execution directory (the directory where ncfs program is executed).

We call the files written into the ncfs-mounted directory as "target files".

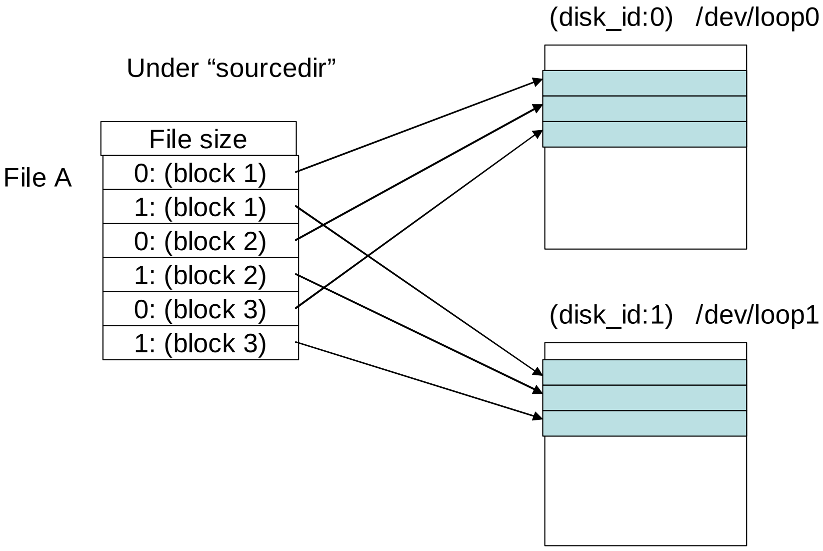
NCFS stores indexes of target files at a "rootdir".

NCFS stores the data of the targets files to the devices in the way stated in the setting files.



<u>Block mapping:</u>

NCFS uses one-to-one block mapping. This is good for implementation of stripped block allocation. Each file stores: (1) file size, and (2) the block mappings.



II. INSTALLATION

<u>(1) Install FUSE and pkg-config</u>

Install FUSE

- ➢ `sudo apt-get install libfuse-dev`

Install pkg-config

- ➢ `sudo apt-get install pkg-config`

(2a) Install NCFS

1. Download and unzip the NCFS package into a directory.
2. Run make in the directory.

(3) Setup and Run NCFS

1. Edit the raid_setting file (Refer to Section III: System settings)

2. Run the setup script: (automatically generated loop devices)

```
> sudo setup.sh
> sudo setup.sh <number of disks> <size of disks>
```

Alternatively, you can setup storage devices (local or networked) manually. Please refer to steps 2(b), 2(c), 2(d).

3. Run the NCFS

To run in debug mode:

```
> sudo ./ncfs -d rootdir mountdir
```

To run in non-debug mode:

```
> sudo ./ncfs rootdir mountdir
```

Unmount the filesystem by running “sudo fusermount -u mountdir”.

(2b) Setup loop devices (optional)

1. Generate a zero-filled file

```
> dd if=/dev/zero of=<file name> bs=1k count=<file size in KB>
```

2. Setup loop device

```
> sudo losetup <loop device name> <file name>
```

For example, to create three loop devices of 1000MB each:

```
> dd if=/dev/zero of=file00.img bs=1M count=1000
> sudo losetup /dev/loop0 file00.img
> dd if=/dev/zero of=file01.img bs=1M count=10000
> sudo losetup /dev/loop1 file01.img
> dd if=/dev/zero of=file02.img bs=1M count=10000
> sudo losetup /dev/loop2 file02.img
```

(2c) Install AoE (optional)

AoE is a protocol enabling access of SATA storage devices over Ethernet.
Non-routable: Storage devices can only be accessed within local network.

1. Install packages

```
(storage)# apt-get install vblade
(initiator)# apt-get install aoetools
```

2. Create AoE devices

```
(storage)# dd if=/dev/zero of=vblade0 bs=1M count=1000
(storage)# vblade 5 1 eth0 vblade0

(initiator)# modprobe aoe
(initiator)# aoe-interfaces eth0
(initiator)# aoe-discover
(initiator)# aoe-stat
```

3. Use AoE devices in NCFS

The name of the AoE device created in this example is: /dev/etherd/e5.1
Just use this device in raid_setting as if it is a local device.
NCFS can use multiple AoE devices, as well as hybrid of local and AoE devices.

(2d) Install iSCSI (optional)

iSCSI allows two hosts to exchange SCSI commands over TCP/IP. Therefore it is routable.

1. Install packages

```
(storage)# sudo aptitude install iscsitarget
(initiator)# sudo aptitude install open-iscsi
```

2. Setup (storage)

Edit “/etc/default/iscsitarget”

```
	Set ENABLE=true
```

Create storage device or partition

Edit “/etc/ietd.conf”

```
	Set global unique name
```

Edit “/etc/initiators.allow”

```
	Set allowed hosts
/etc/init.d/iscsitarget start
```

For further details please refer to:
http://www.howtoforge.com/using-iscsi-on-ubuntu-10.04-initiator-and-target

3. Setup (initiator)

Edit “/etc/iscsi/iscsid.conf”

```
	Set node.startup=automatic
/etc/init.d/open-iscsi restart
iscsiadm –m discovery –t st –p <target IP>
iscsiadm –m node
iscsiadm -m node --targetname “<global unique name>" --portal “<target IP>:3260"
--op=update --name node.session.auth.authmethod --value=CHAP
iscsiadm -m node --targetname “<global unique name>" --portal “<target IP>:3260"
--op=update --name node.session.auth.username --value=someuser
iscsiadm -m node --targetname “<global unique name>" --portal “<target IP>:3260"
--op=update --name node.session.auth.password --value=secret
/etc/init.d/open-iscsi restart
fdisk –l <device name>
	E.g. sudo fdisk –l /dev/sdb
```

For further details please refer to:
http://www.howtoforge.com/using-iscsi-on-ubuntu-10.04-initiator-and-target

III. SYSTEM SETTINGS

The followings are the values used in the setting files:

Disk raid type:

Number := Type
100 := jbod
0 := raid0
1 := raid1
4 := raid4
5 := raid5
6 := raid6
1000 := mbr (exact minimum bandwidth regenerating code)
3000 := reed-solomon code

Disk status:

0 := normal
1 := fail (cannot open)

Operation mode:

0 := normal
1 := degraded (read only)
2 := incapable (cannot read and write)

There are three system setting files:

1. raid_setting
2. raid_metadata
3. raid_health

raid_setting: defines the file system settings

Syntax:

```
disk_total_num=<number of disks>
data_disk_num=<number of disks used for storing native data>
disk_block_size=<size of each block in bytes>
disk_raid_type=<type of raid>
space_list_num=0 <for deletion, currently not used>
<disk_id>.device name = <device name>
<disk_id>.free_offset = 0
<disk_id>.free_size=<size in bytes>
```

An example:

```
disk_total_num=3
data_disk_num=2
disk_block_size=4096
disk_raid_type=5
space_list_num=0
0.dev_name=/dev/loop1
0.free_offset=0
0.free_size=15728640
1.dev_name=/dev/loop2
1.free_offset=0
1.free_size=15728640
2.dev_name=/dev/loop3
2.free_offset=0
2.free_size=15728640
```

raid_metadata: records the metadata of the file system

This file is used by the NCFS to record the available spaces of the disks. The data are stored in binary format.

raid_health: records the status of the disks

This file records the disk status: 0 for health; 1 for failed.

Syntax:

```
<status of disk id 0>
<status of disk id 1>
…
```

IV. RECOVERY

NCFS has two recovery tools: recovery, and remap.

1. Recovery

"Recovery utility" is used for recovery for RAID 1, 4, 5, 6; and MBR (exact reapir).

An example of using recovery:

Raid setting:

    Type: Raid 5;

Number of devices: 3

Raid health (Edit raid_health):

Device 1: fail; device 2: on; device 3: on.

Recovery:

Run “sudo ./recovery <target device or file>”

This would generate recovered data on the <target device or file>.

Checking:

Replace the failed device/file by the <target device or file>.

Check if the content of the raid storage is the same as before.

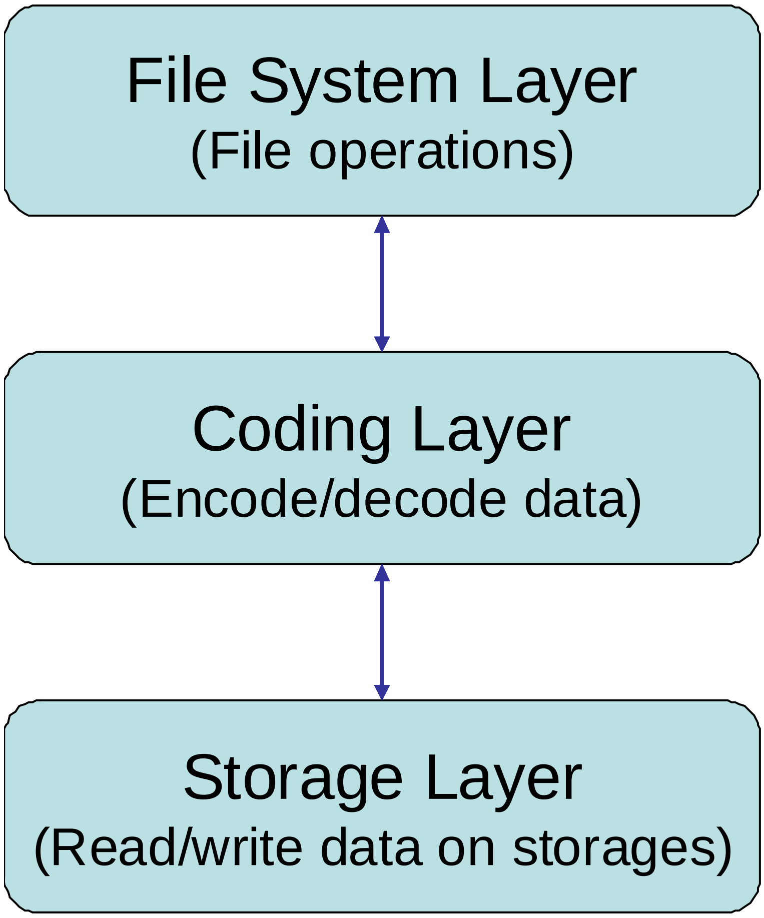
2. Remap

“Remap utility” is used for re-allocate disk data for RAID 5 after there is disk failure. This can restore the system’s operation mode from degraded mode to normal mode.

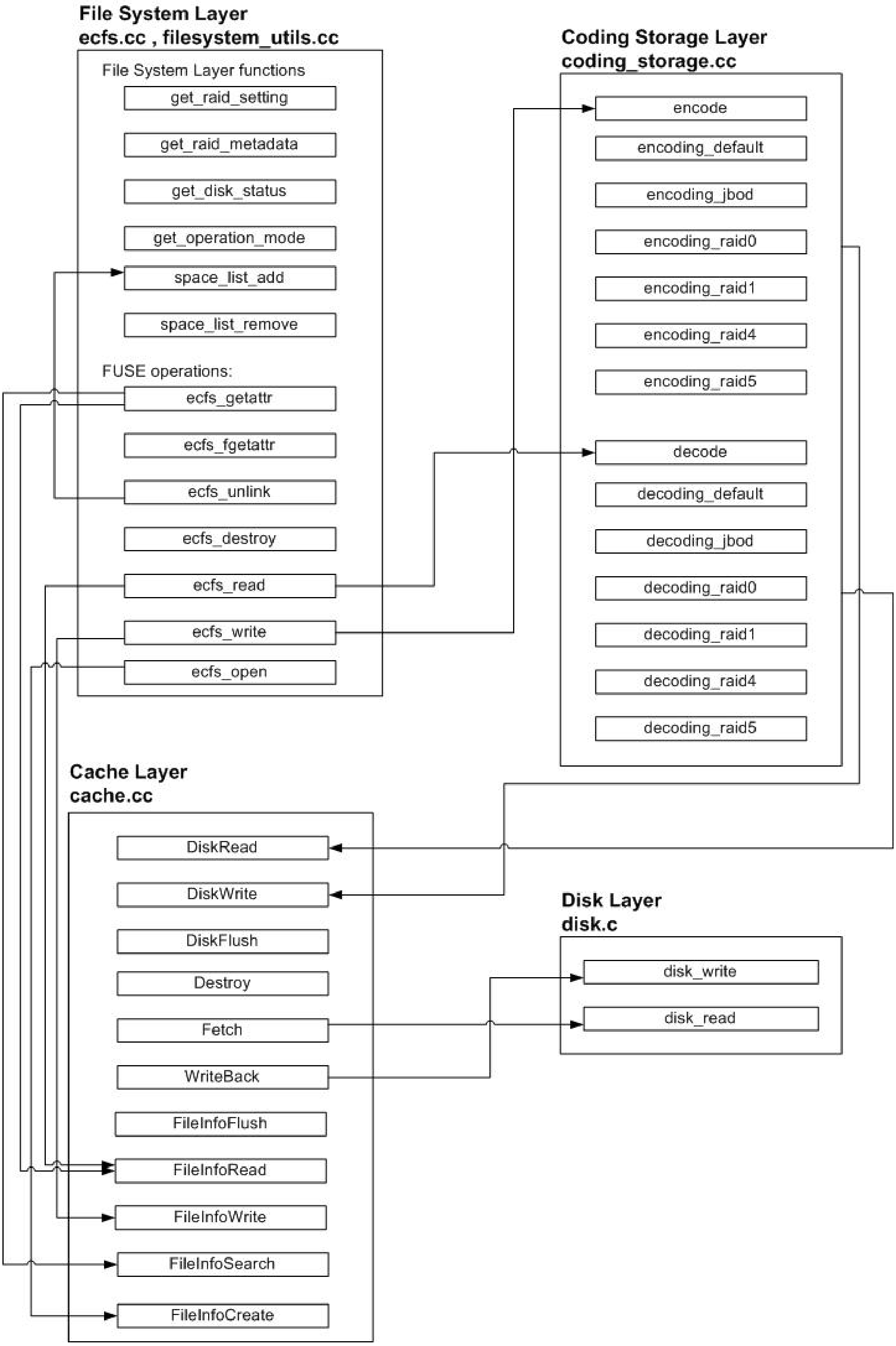
To run remap:

```
> sudo ./remap
```

V. SYSTEM LAYERS



VI. PROGRAM STRUCTURE

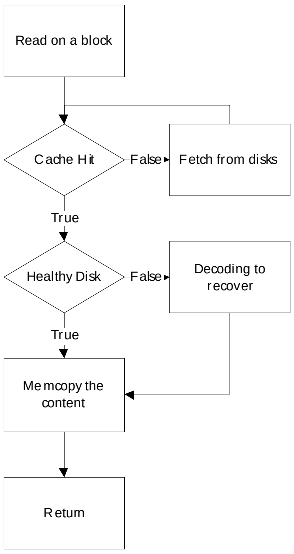


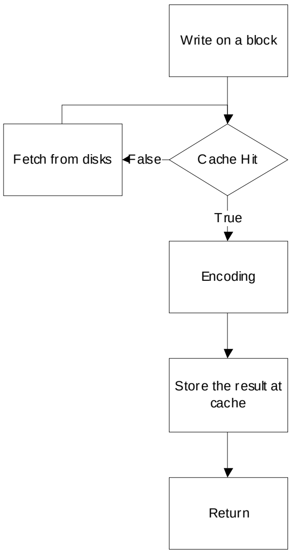
VII. DIRECTORY LISTING

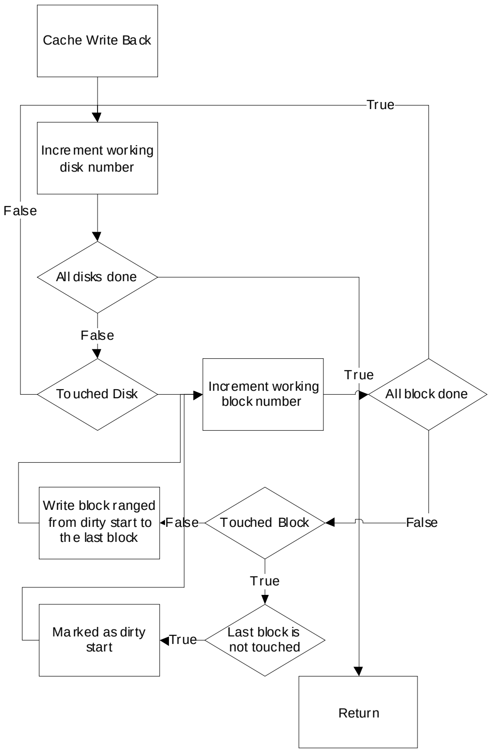
/src	(source files)
/doc	(documentation files)
/settings_template	(templates of settings)
/src/filesystetm	(file system layer)
/src/cache	(cache layer)
/src/storage	(storage layer)
/src/coding	(coding layer)
/src/network	(network layer)
/src/gui	(GUI)
/src/utility	(utility programs, e.g. recovery programs)
/src/jerasure	(Jerasure library)

VIII. MECHANISMS

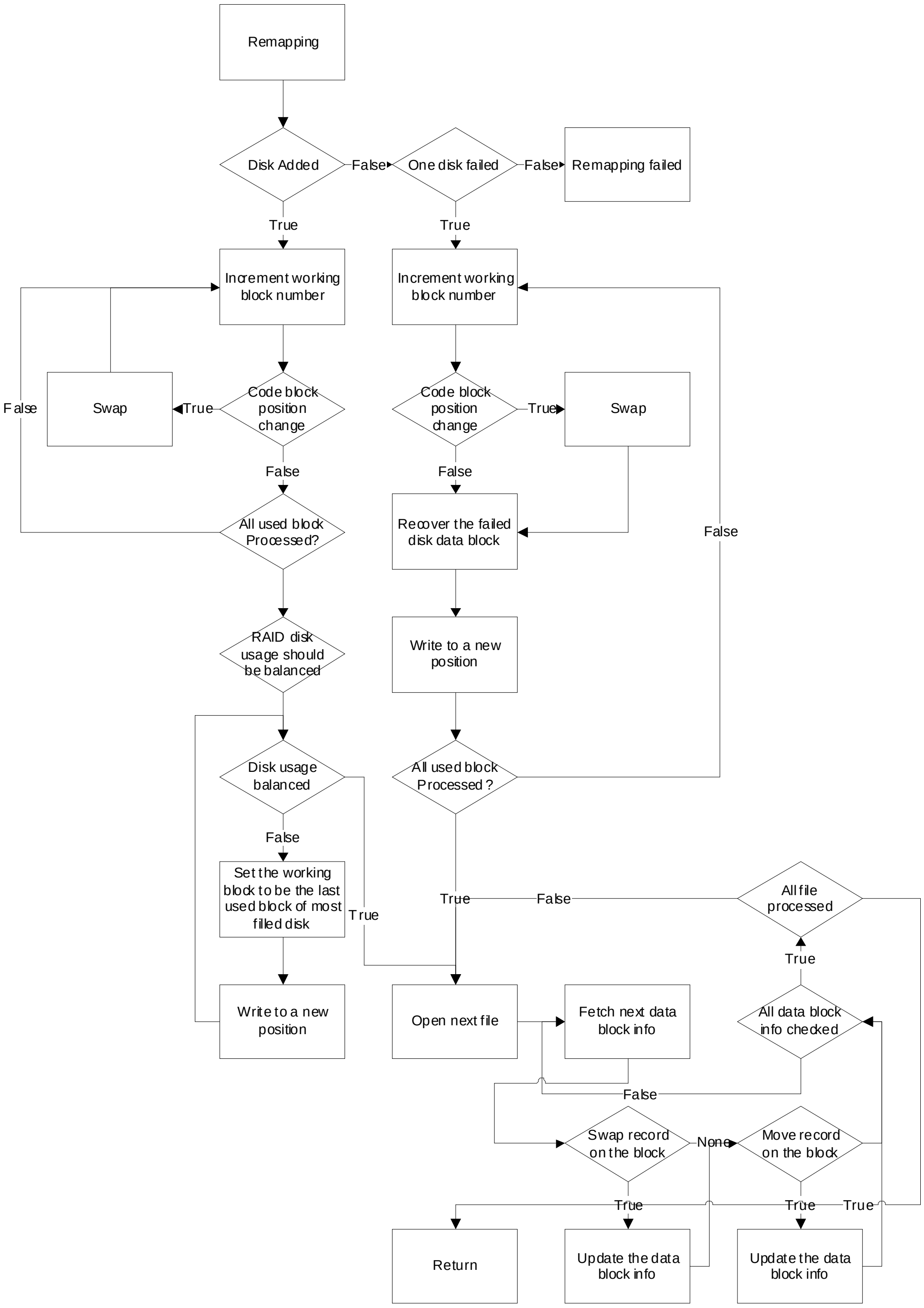
Cache principle:






Cache Write Back
True
Increment working disk number
False
All disks done
False
True
Touched Disk
Increment working block number
All block done
Write block ranged from dirty start to the last block
False
Touched Block
False
True
Marked as dirty start
True
Last block is not touched
Return

Remapping principle


Remapping
Disk Added
False
One disk failed
False
Remapping failed
True
True
Increment working block number
Increment working block number
False
Swap
True
Code block position change
Code block position change
True
Swap
False
False
All used block Processed?
Recover the failed disk data block
False
RAID disk usage should be balanced
Write to a new position
Disk usage balanced
All used block Processed ?
False
Set the working block to be the last used block of most filled disk
True
True
False
All file processed
True
Write to a new position
Open next file
Fetch next data block info
All data block info checked
False
Swap record on the block
None
Move record on the block
True
True
True
Return
Update the data block info
Update the data block info