NCFS Developer's Guide

NCFS version: 1.2.0

Date: March 2012

Authors:

Hu Yuchong (ychu@inc.cuhk.edu.hk)

Lee Pak Ching (pclee@cse.cuhk.edu.hk)

Li Runhui (rhli@cse.cuhk.edu.hk)

Li Yan Kit (ykli7@cse.cuhk.edu.hk)

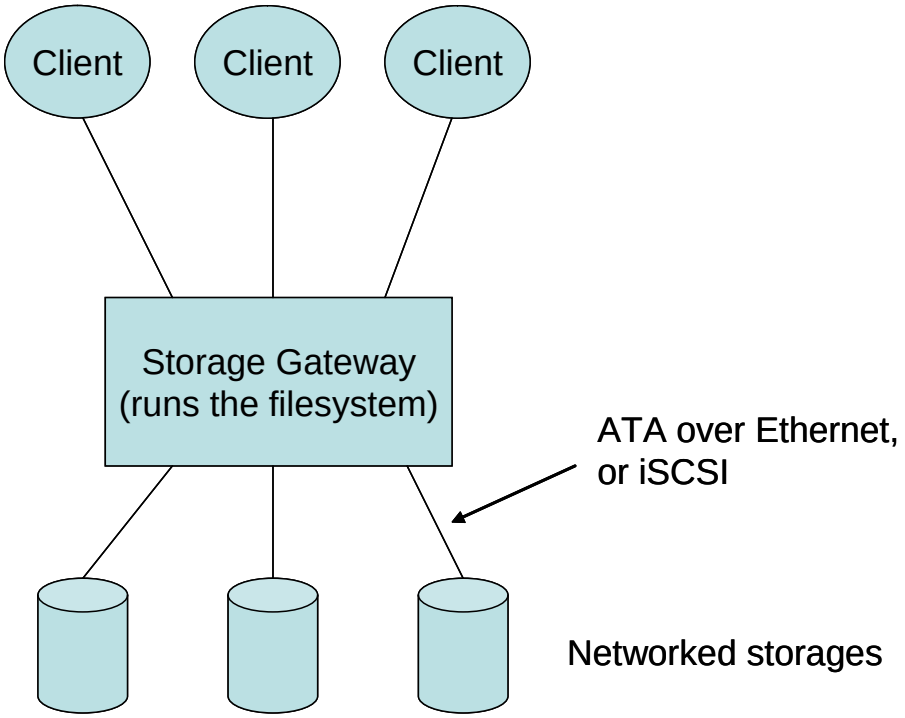
Yu Chiu Man (cmyu@cse.cuhk.edu.hk)

Zhu YunFeng (zyfl@mail.ustc.edu.cn)

Contents:

I. INTRODUCTION

NCFS (Network Coding File System) is a distributed file system supporting various regenerating and erasure coding schemes. NCFS can connect with networked storage devices through ATA over Ethernet or iSCSI. It can encode and decode the data on the storages according to customizable coding modules.



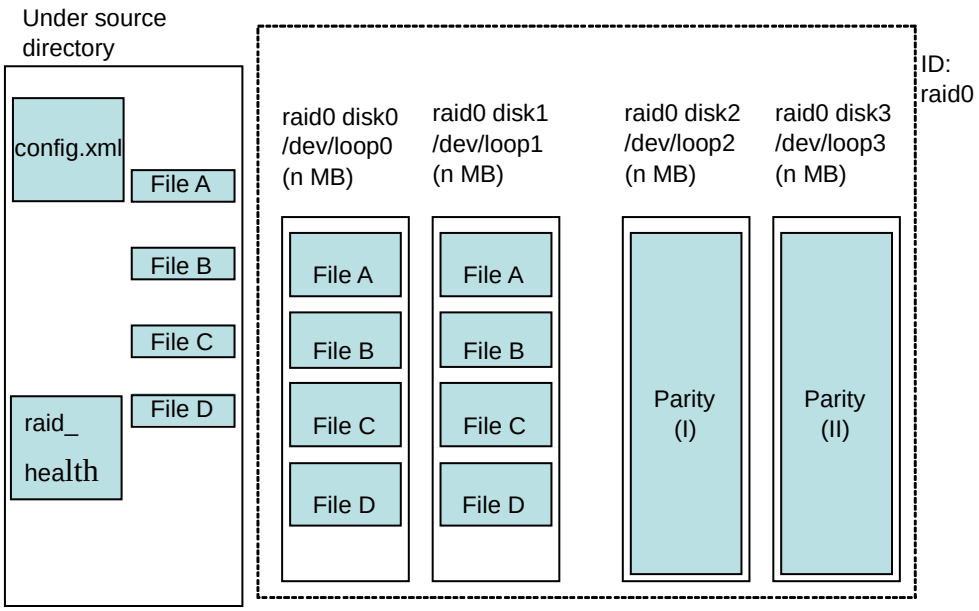
Disk layout:

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”.

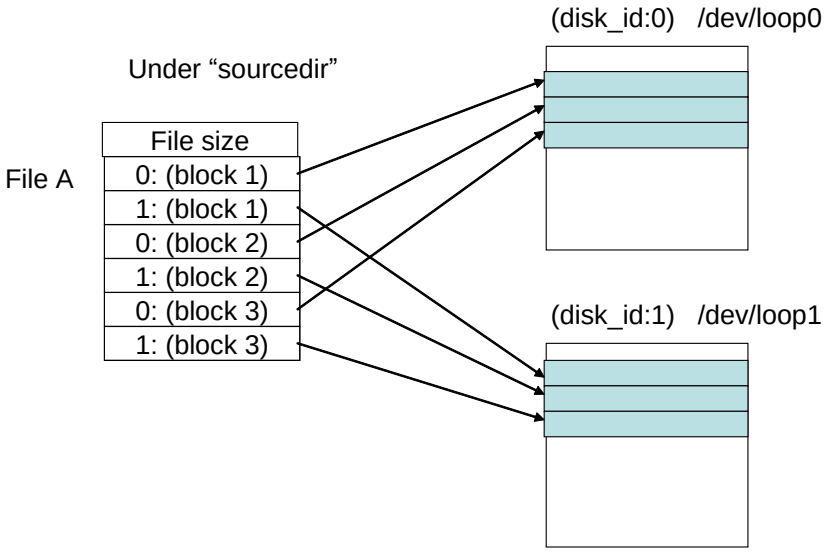
NCFS stores indexes of target files at a “rootdir”.

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



Block mapping:

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

(1) Install FUSE and pkg-config

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 config.xml 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 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)
13 := reed-solomon code
130 := src-rs (Simple Regenerating 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 two system setting files:
1. config.xml
2. raid_health

config.xml: defines the file system settings

Syntax:

Element	Example	Description
<NoCache>	1	use cache or not
<NoGui>	1	use Gui or not
<SrcF>	2	Parameter F of SRC
<Experiment>	0	record breakdown time or not
<TotalDiskNumber>	5	number of disks
<DataDiskNumber>	4	number of disks used for storing native data
<ChunkSize>	131072	size of each block in bytes
<RaidType>	5	type of raid
<DevName>	/dev/loop1	device name
<TotalSize>	268435456	size of each disk in bytes
<FreeOffset>	0	size in bytes
<FreeSize>	268435456	size in bytes

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 a recovery tool.

1. Recovery

“Recovery utility” is used for recovery for RAID 1, 4, 5, 6; MBR (exact reapir) and Reed-Solomon Code.

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 ./recover <target device or file> <id of the failed device>”

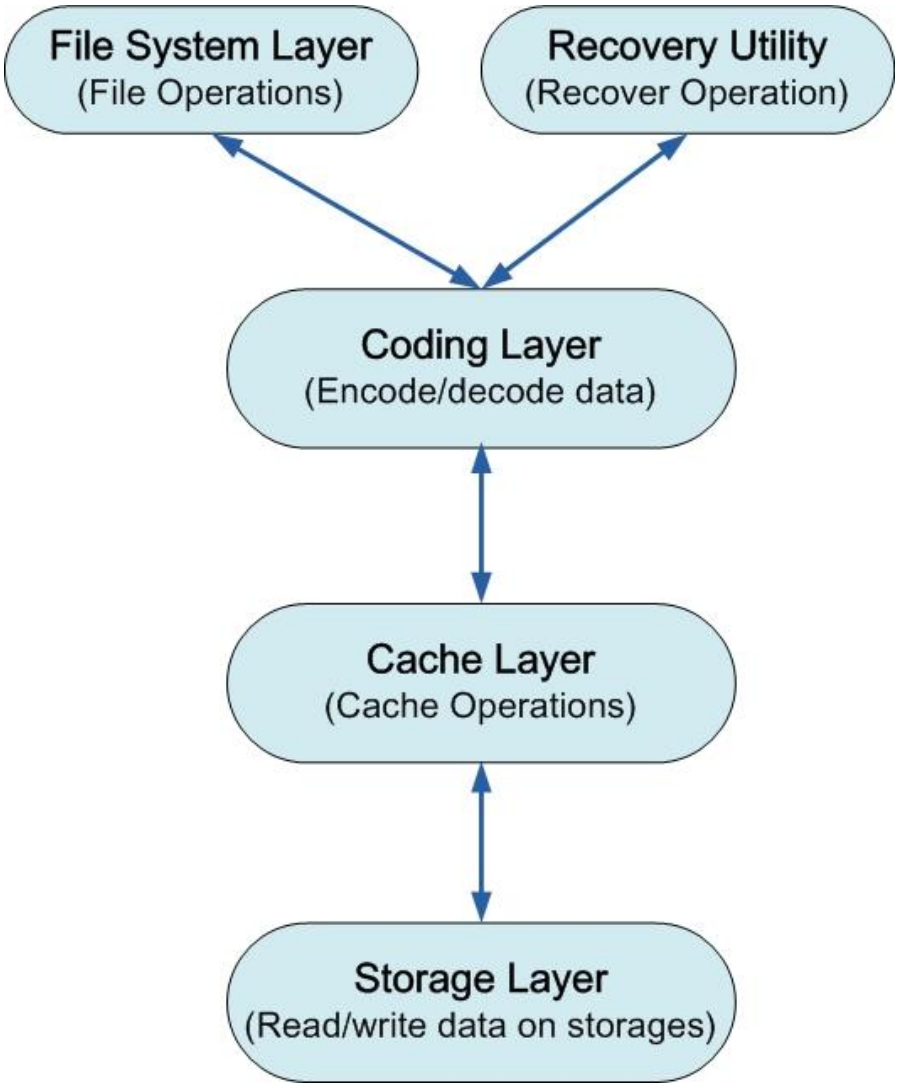
This would generate recovered data on the <target device or file>.

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.

V. SYSTEM LAYERS



The file system layer, the coding layer, and the storage layer are defined in our NetCod'11 paper (see http://ansrlab.cse.cuhk.edu.hk/software/ncfs).

The cache layer is *not* being used right now.

The recovery utility is an independent utility for the recovery operation.

VI. 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/config	(operations on config.xml)
/src/network	(network layer)
/src/gui	(GUI)
/src/utility	(utility programs, e.g. recovery programs)
/src/jerasure	(Jerasure library)

VII. LIMITATIONS

Currently, we do not consider the overflow case. Writing to a fully-filled storage node may create unexpected problems.

For a given chunk size C (specified in config.xml), we currently support the file size that is either a multiple of C or not a multiple of 4KB. For example, if C = 512KB, then we support the file size that is equal to 1024KB, 2048KB, or 1025KB, but we don’t support 516KB. Note that if C = 4KB, we support all file sizes.

Disk usage on SRC is inaccurate