DCFS v1.0 User Guide

Jin Xiong, Si-Ning Wu
National Research Center for Intelligent Computing Systems,
Institute of Computing Technology, Chinese Academy of Chinese
November 15, 2003

ABSTRACT

This document first introduces features of DCFS, then describes architecture and software components of DCFS. From section 2 to 4, steps to compile, install, configure, deploy, start up and shut down DCFS are illustrated. In section 5, failure process is given.

1. Introduction to DCFS

DCFS is a cluster file system developed by NCIC (National Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Chinese) for Linux clusters, especially the Dawning 4000L Super-server. It is a shared global file system with single system image. Applications see a unified name space from all cluster nodes. Files in DCFS can be accessed through standard system calls, such as ***open, close, read, write,*** etc, and can be manipulated by system commands such as ***ls, cp, mkdir, rmdir, tar, vi,*** etc. Applications using traditional UNIX file I/O can use DCFS files without recompilation.

DCFS exploits a scalable architecture as shown in Figure 1. Metadata processing is divorced from file data processing in DCFS. They have different servers and different access path. Moreover, there can be multiple servers for both file data and metadata. Such architecture eliminates the performance bottleneck in single server systems.

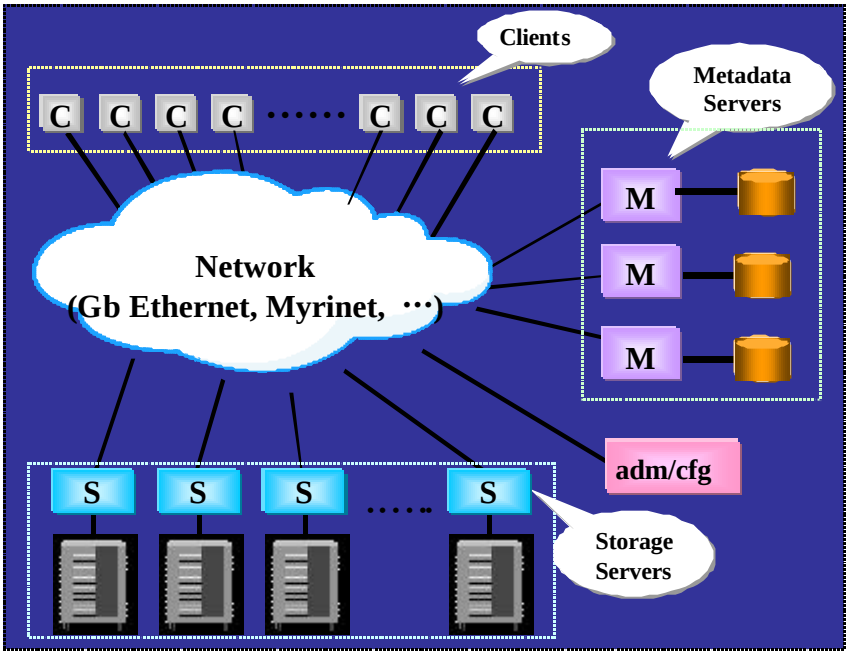

Figure 1 The Architecture of DCFS

File data are stored on multiple storage servers in a RAID 0 style of striping, and can be accessed concurrently. Metadata are stored on and maintained by multiple metadata servers. DCFS' metadata distribution policy enables the simultaneous metadata processing on multiple metadata servers. Both metadata and file data are stored in regular files of native file system (EXT2) on servers.

DCFS provides a set of management utilities. Their functions include DCFS system configuration, starting and stopping DCFS services, and mount and unmount DCFS file system.

1.1. Features

- Shared global file system for Linux clusters with single system image
- Standard interface: OS system calls and system commands
- Parallel data accesses on multiple storage servers
- Parallel metadata processing on multiple metadata servers
- High scalability
- High aggregate I/O bandwidth

- High metadata performance
- Easy management

1.2. System overview

As shown in Figure 1, cluster nodes are classified into 4 types according to the roles they play in DCFS.

- Storage servers: Nodes that store DCFS file data are called storage servers. Usually these nodes have directly accessed, high-performance RAIDs so that they have much high disk I/O bandwidth. File data are stored on these storage servers in a RAID 0 style of striping.
- Metadata servers: DCFS metadata information are called metadata servers. Metadata managed by metadata servers include the metadata of files and directories（such as file size, owner, access mode, access date, and other attributes）, file data layout information, directory files and superblock。 DCFS supports to create multiple DCFS file systems, which are known as logical volumes in DCFS. Each DCFS file system has its own metadata servers, and one of which is called super-manager。 The super-manager stores and maintains he file system superblock, root directory and root attributes, while each of the others stores and maintains a subtree of the root directory.
- Client nodes: Nodes that mount DCFS file system to use are called client nodes.
- The adm/cfg node: the node that runs the Administration Agent is called the adm/cfg node. This node maintains the configuration information and all logical volumes information.

Compute nodes are used for CPU-intensive and communication-intensive computing, while storage nodes are used to service intensive disk accesses. Clusters now tend to have dedicated storage nodes. DCFS is developed with the same intension. In DCFS, server nodes including storage server nodes and metadata server nodes cannot be client nodes. However, a server node can be a storage server node, or a metadata server node, or both a storage server node and a metadata server node. A typical configuration for DCFS on a cluster system is that compute nodes are configured as client nodes, dedicated storage nodes are configured as storage server nodes, metadata servers are configured on some secure nodes and the adm/cfg node is configured on the console node.

1.3. DCFS Components

There are six major components to the DCFS system:

- mgr: It is a user level daemon running on each metadata server.
- ios: It is a user level daemon running on each storage server.
- clerk: It is a user level daemon running on each client node.
- broker: It is a kernel module to be inserted on each client node.
- cnd and scnd: They are user level daemons. cnd runs on the adm/cfg node, while scnd runs on the backup adm/cfg node for high availability considerations.
- dcfs_adm and dcfs_cfg: They are administration utilities for DCFS management.

As shown in Figure 2, broker is a kernel module that implements the VFS interface. This implementation enables applications using system calls interface to access DCFS files. So application programs based on traditional UNIX I/O can use DCFS files without modification or recompilation.

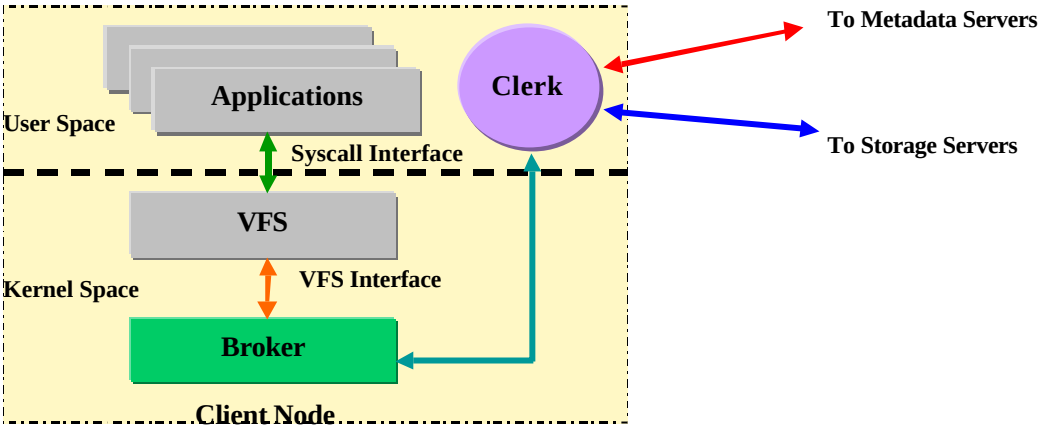

Figure 2 Data Flow

Current version of DCFS is implemented and tested on Linux-2.4.18-3smp。If you need to run DCFS on kernels other than Linux-2.4.18-3smp, broker codes need some adjustments.

2. Installation and Configuration

Note: make sure that rexec and rcp commands are executed as root user on all related nodes.

2.1. Obtaining the source

DCFS is freely available on web. You can get the source code package on web site: http://www.ncic.ac.cn/dcfs/ or send mail to dcfs@ncic.ac.cn. These files are tar archives of the DCFS source which have subsequently been compressed with the ***gzip*** tool (dcfs-1.0.tar.gz) or ***compress*** tool (dcfs-1.0.tgz). At the time of writing the newest version of the DCFS archive was 1.0.0 which can run on Linux platform with kernel version 2.4.x.

2.2. Compiling the package

First, untar DCFS source code package. For dcfs.tar.gz, enter commands:

[root@node1 dcfs] gzip –d dcfs.tar.gz

[root@node1 dcfs] tar xf dcfs.tar

For dcfs.tgz:

[root@node1 dcfs] tar -zxf dcfs.tgz

DCFS source code package includes such directories: *conductor, broker, clerk, MGR, IOS, COM, help, include, tools, FSCK* and three files—*Makefile, Makefile.conf* and *COPYRIGHT*. The files in source code package are listed in appendix A.

Configure the Makefile.conf to set the correct source code directory, and enter the command to compile DCFS:

[root@node1 dcfs] make

2.3. Installation

DCFS use /etc/dcfs as the default installation directory. Before Installation, make sure that the /etc/dcfs , /etc/dcfs/bin, /etc/dcfs/conf directories have been created. Run the command in dcfs directory:

[root@node1 dcfs] make install

Now all executable binary files are copied to /etc/dcfs/bin. These files include dcfs.o, clk, clerk, ios, mgr, cnd, scnd, dcfs_adm, dcfs_cfg, etc.

Note: “make install” only install the files in local node. To distribute the files to other nodes, user must execute commands described in section 2.4.

2.4. Configuration

DCFS places configuration files on three sub-directories under /etc/dcfs/conf. These sub-directories are *configuration, input-files and logical-volumes.*

A global configuration file dcfs.conf is in sub-directory *configuration*. It contains the information of all client nodes, metadata server nodes, storage server nodes and configuration agent node. It also includes the communication protocol of DCFS (such as TCP/IP). Figure 3 gives a example of dcfs.conf. Remark line begins with #. Note: currently, a client node can't configure as server node.

```
com -> TCP

clerk:
11.18.1.0
11.18.1.1
11.18.1.2
11.18.1.3
11.18.1.4
11.18.1.5
11.18.1.6
11.18.1.7
11.18.1.8
11.18.1.9
11.18.1.10
11.18.1.11
11.18.1.12
11.18.1.13
11.18.1.14
11.18.1.15
11.18.2.32
11.18.2.33
11.18.2.34
11.18.2.35
11.18.2.36
11.18.2.37
11.18.2.38
11.18.2.39
11.18.2.40
11.18.2.41
11.18.2.42
11.18.2.43
11.18.2.44
11.18.2.45
11.18.2.46
11.18.2.47

ios:
11.18.3.24
11.18.3.25
11.18.3.26
11.18.3.27

mgr:
11.18.3.28
11.18.3.29
11.18.3.30
11.18.3.31

cnd:
11.18.1.0
11.18.1.1
```

Figure 3. dcfs.conf

The configuration file new-lv.conf to create a new volume locates in sub-directory *input-files*. It contains the information to create a file system volume: super metadata server, list of storage servers and list of metadata server. When creating a volume, DCFS would allocate a volume number, such as lv00, lv01. Figure 4 is a example of new-lv.conf. When successfully create a volume, DCFS put a configuration file lvxx.conf in sub-directory *logical-volumes*.

```
#first line means using auto or manual method;
#1 means auto, 0 means manual;

AUTO                    =>      0
PV-NUM                  =>      4
METANODE-NUM            =>      4
SUPER-MGR               =>      11.18.3.28

#the layout of this new volume, including
#ios location and disk number;

PV1             =>      11.18.3.24      /dev/sda3
PV2             =>      11.18.3.25      /dev/sda3
PV3             =>      11.18.3.26      /dev/sda3
PV4             =>      11.18.3.27      /dev/sda3


MetaNode1       =>      11.18.3.28
MetaNode2       =>      11.18.3.29
MetaNode3       =>      11.18.3.30
MetaNode4       =>      11.18.3.31
~
~
```

Figure 4 new-lv.conf

These configuration files can be modified manually and created using dcfs_cfg. Dcfs_cfg is to create dcfs.conf, new-lv.conf and some shell scripts, including dcfs_prsh, dcfs_clean, dcfs_mount, dcfs_umount, dcfs_lsmount and dcfs_rm. Note: you must make sure that rexec and rcp commands are executed as root user on all related nodes.

Dcfs_prsh is a simple tool to execute shell commands on all DCFS nodes. For example, the command "dcfs_prhs hostname" would run the hostname command on all DCFS nodes. The script dcfs_clean would umount DCFS, kill the daemons and uninstall kernel modules. This tool is used when DCFS can't normally starting up or shutting down. The scripts dcfs_mount and dcfs_umount is used to mount and unmount all DCFS volumes on all client nodes. The tool dcfs_rm is used only when clean up DCFS and it would remove all DCFS system data. All these script can be modified manually. In appendix B, we give the example of these shell script tools.

Run the dcfs_cfg on the node which has installed executable code. The user interface of dcfs_cfg is illustrated in Figure 5.

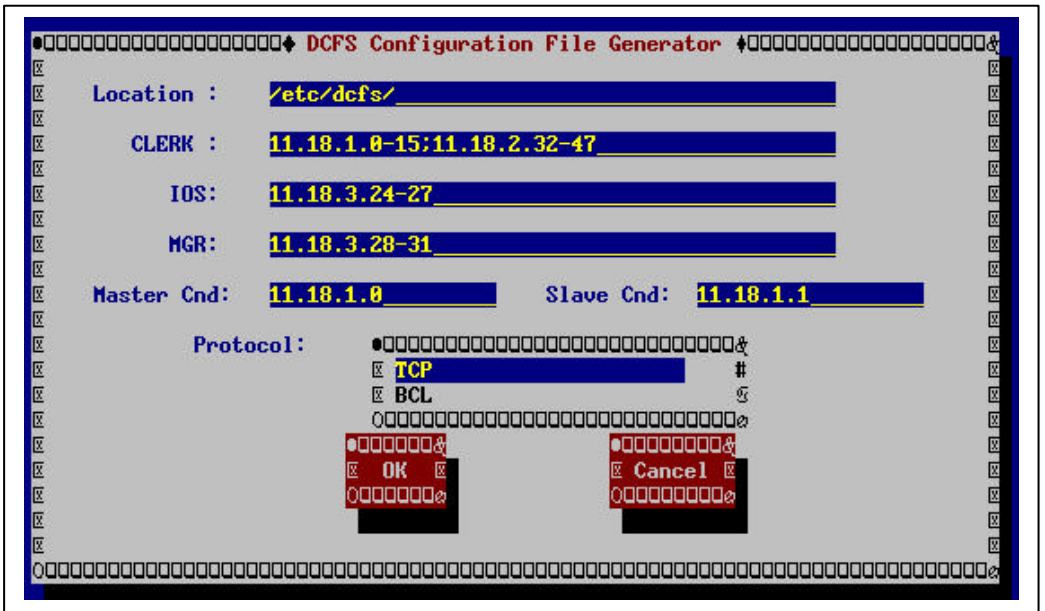

Figure 5 dcfs_cfg user interface

- "Location" field sets the directory where to put the configuration file and shell scripts.

This field must be /etc/dcfs/bin. You can change this directory only when you want to check the configuration files.

- "CLERK" filed configures the set of client nodes. Current version dcfs_adm uses IP address to identify a node. ";" is used when two IP address is same except the last byte. "-" is used to denote continuous IP address. For example, "11.18.1.0-1;4-5" includes four nodes: 11.18.1.0, 11.18.1.1, 11.18.1.4 and 11.18.1.5.
- "MGR" filed sets DCFS metadata server (mgr) nodes. The rule of expression is as same as "CLERK" field.
- "IOS" filed sets DCFS storage server (ios) nodes. The rule of expression is as same as "CLERK" field.
- "Master cnd" filed sets DCFS's master configuration agent (one node).
- "Master cnd" filed sets DCFS's slave configuration agent (one node)
- "Protocol" field sets the communication protocol used by DCFS. Currently, DCFS only support TCP/IP.

After filling the fields above, choose the "OK" button and press "Enter" key. A note window will pop up, press "Enter" key, then a line **"** Do you want to copy the configuration file to all the nodes:[y]" shows up, pressing "Y" would copy the configuration files to all DCFS nodes.

Note 1: DCFS stores all metadata in directory /dcfs/meta which is defined in "dcfs/MGR/include/mgr.h". You can change this directory by modifying the header file and re-compiling mgr.

Note 2: DCFS stores all file data in directory /dcfs/data which is defined in "dcfs/IOS/include/ios.h". You can change this directory by modifying the header file and re-compiling ios.

2.5. Deployment

After configuring DCFS, run dcfs_prsh to distribute DCFS executable code to all nodes. The command is illustrated below:

```
[root@node1 bin] hostname
node1
[root@node1 bin]dcfs_prsh rcp –r node1:/etc/dcfs /etc
```

3. Start up and Shut down

DCFS provides dcfs_adm tool to start up and shut down all software components. Before starting up DCFS, you must make sure that: (1)DCFS executable binary files are distributed to correct directory on all nodes; (2) configuration files, such as dcfs.conf and new-lv.conf have been correctly created; (3) You can execute ***rexec*** and ***rcp*** as root user on all nodes of DCFS.

3.1. Start up

Starting up DCFS includes three steps: (1) Start up the configuration agent ***cnd;*** (2) Start up all servers, including storage servers (***ios***) and metadata servers (***mgr***); (3) Install kernel modules and DCFS client daemons (***clerk***) on all client nodes. If there isn't any file system volume being created before, you must create a volume using dcfs_adm after executing the starting up steps described above. If you have created volumes, DCFS software components would get volumes' information after starting up. Now you can mount DCFS on all client nodes. DCFS provides a tool (dcfs_mount) to help mount DCFS file system volumes.

To shut down DCFS, you must first unmount DCFS on all client nodes. The procedure of shutting down also includes three steps: (1) Shut down all ***clerks*** and uninstall kernel modules; (2)Shut down all ***mgr***s and ***ios***es. (3) Shut down the configuration agent cnd. Note: now DCFS can only stop all volumes' servers rather than a volume's servers.

Execute dcfs_adm in /etc/dcfs/bin, and it would pop up following user interface. You can

start up DCFS as follows:

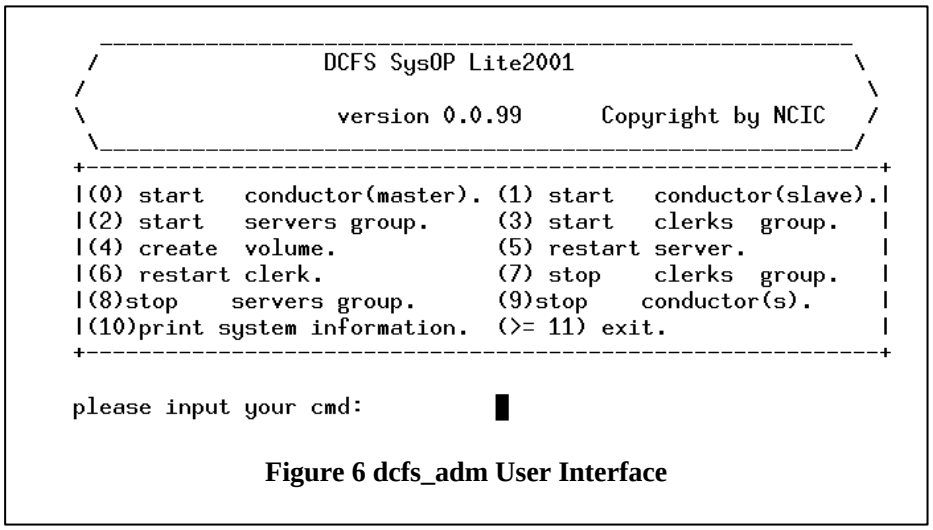

Figure 6 dcfs_adm User Interface

1. Execute dcfs_adm and input 0; # Start up cnd
2. Execute dcfs_adm and input 2; # Start up all ioses and mgrs
3. Execute dcfs_adm and input 3; # Start up all clerks and install kernel modules
4. Execute dcfs_adm and input 4, # Create a volume only when first start up DCFS
5. Execute dcfs_mount, # Mount DCFS on all client nodes

Now, DCFS starting up is ok and you can run applications on DCFS freely.

3.2. Shut down

Before shutting down DCFS, you must make sure that DCFS is running normally. You must execute following steps to shut down DCFS:

1. Execute dcfs_mount
2. Execute dcfs_adm and input 7; # Shut down clerks and uninstall all kernel modules
3. Execute dcfs_adm and input 8; # Shut down all ioses and mgrs
4. Execute dcfs_adm and input 9; # Shut down cnd

Note: If any error occurs during starting up or shutting down DCFS, the followed steps would fail and you must process the error using the method described in section 6.

4. Applications

DCFS supports most of POSIX system calls and RMIO, so most of regular applications running on UNIX can access DCFS without being re-complied and the parallel applications using RMIO can also run on DCFS.

4.1. Regular Applications

The regular application in DCFS is referred to all programs running on one node (including single process, multi-process and multi-thread programs, such as mailserver, httpd) and programs running on multi-nodes which use POSIX system calls. The regular applications can operate DCFS files through OS system command, system call and library functions.

DCFS support following system commands: mount、unmount、df、find、ls、mkdir、rmdir、rm、mv、cp、vi、cat、tar etc.

DCFS support following POSIX system calls: read、write、open、close、mkdir、rmdir、rename、flock etc；

DCFS also support library functions: lseek、fopen、fclose、fget、fput、fprintf etc.

4.2. Parallel Applications

DCFS supports parallel applications which access file through MPI-IO interface. RMIO can run on UNIX file systems(such as ext2) which provide standard POSIX system calls, and DCFS provides the same system calls as ext2. So RMIO can run on DCFS. We have run two parallel programs based RMIO on DCFS: Flash I/O benchmark and MM5.

5. Failure Process

DCFS provides FSCK tool to ensure that the namespace is consistence when an ***mgr*** node is down. But DCFS currently doesn't ensure that the data on storage server is correct. To ensure that DCFS can still run after a software component is down, we give following suggestions: (1) Backup DCFS system data periodically, including metadata on metadata server mgr and file data on storage server ios; (2) When a mgr is down, use FSCK to check the consistency of DCFS namespace and then restart the mgr. Note: Currently, DCFS FSCK can only ensure the consistency of namespace **when a mgr is down.**

5.1. How to handle failure during running an application

The first step is to find out the node which occurred failure and kill the application, then restart the DCFS according to the type of node.

1. If the down node is a client node. Run the dcfs_adm to restart DCFS and then mount DCFS. Note: current version DCFS doesn't checks whether the namespace is consistency. We will implement this function in next version. If you find some incorrect files in DCFS, you can remove these files manually.
2. If the down node is a storage server. Run the dcfs_adm to restart storage server.
3. If the down node is an mgr node. When the node is restart, run FSCK in this node and then restart the mgr using dcfs_adm. In section 5.3, we show how to use FSCK to restart an mgr. If occur fatal error during executing FSCK, umount DCFS on all client nodes and run dcfs_adm to shut down DCFS (if failed, using dcfs_clean to shut down all components). Restore the backup system data and then restart DCFS.

5.2. How to handle failures during starting up

If dcfs_adm can't start up DCFS, it shows that DCFS is crashed. You can handle this case as follows:

1. Run dcfs_clean to shut down all DCFS components.
2. Run dcfs_cleanup to clean up all DCFS system data.
3. Restore the most newly backup data.
4. Run dcfs_adm to start up DCFS.
5. Mount DCFS on all client nodes.

5.3. FSCK

When an mgr down, users can use FSCK to check the consistency of DCFS namespace. After the node restart, run FSCK as follows:

1. Copy all logic volume information (lvxx.conf) to the directory FSCK exists.
2. Run FSCK –v vol_no –c lvxx.conf, vol_no is the volume's number to check and lvxx.conf is its corresponding configuration file.
3. If all are ok, run dcfs_adm to start up DCFS as described in section 3.1.
4. If there is any fatal error, restart DCFS following the step in section 5.2.

Appendix A: Publication

1. *Jin Xiong, Sining Wu, Dan Men, Ninghui Sun, and Guojie Li*, **Design and Performance of the Dawning Cluster File System**, 2003 IEEE International Conference on Cluster Computing, December 1-4, 2003, Hong Kong

2. *Rongfeng Tang, Dan Mend, Sining Wu*, **Optimized Implementation of Extendible Hashing to Support Large File System Directory**, 2003 IEEE International Conference on Cluster Computing, December 1-4, 2003, Hong Kong
3. Jin He, Jin Xiong, Sining Wu, Yi Lu, Dan Men, **DCFS: File Service in Commodity Cluster Dawning4000**, The 4th International Parallel and Distributed Computing, Applications and Technologies, August 27-29, 2003, Cheng Du, P.R.China

Appendix B: DCFS Configuration files

B.1 dcfs.conf

```
com -> TCP

clerk:
11.18.1.0
11.18.1.1
11.18.1.2
11.18.1.3
11.18.1.4
11.18.1.5
11.18.1.6
11.18.1.7
11.18.1.8
11.18.1.9
11.18.1.10
11.18.1.11
11.18.1.12
11.18.1.13
11.18.1.14
11.18.1.15
11.18.2.32
11.18.2.33
11.18.2.34
11.18.2.35
11.18.2.36
11.18.2.37
11.18.2.38
11.18.2.39
11.18.2.40
11.18.2.41
11.18.2.42
11.18.2.43
11.18.2.44
11.18.2.45
11.18.2.46
11.18.2.47

ios:
11.18.3.24
11.18.3.25
11.18.3.26
11.18.3.27

mgr:
11.18.3.28
11.18.3.29
11.18.3.30
11.18.3.31

cnd:
11.18.1.0
11.18.1.1
```

B.2 new-lv.conf

```
#first line means using auto or manual method;
#1 means auto, 0 means manual;

AUTO                    =>      0
PV-NUM                  =>      4
```

```
METANODE-NUM          =>      4
SUPER-MGR             =>      11.18.3.28

#the layout of this new volume, including
#ios location and disk number;

PV1             =>      11.18.3.24      /dev/sda3
PV2             =>      11.18.3.25      /dev/sda3
PV3             =>      11.18.3.26      /dev/sda3
PV4             =>      11.18.3.27      /dev/sda3


MetaNode1       =>      11.18.3.28
MetaNode2       =>      11.18.3.29
MetaNode3       =>      11.18.3.30
MetaNode4       =>      11.18.3.31
```

B.3 dcfs_prsh

```
#!/bin/bash

if [ $# -ne 0 ]
then
echo rsh 11.18.1.0 $* ...
rsh 11.18.1.0 $*
echo

echo rsh 11.18.1.1 $* ...
rsh 11.18.1.1 $*
echo

echo rsh 11.18.1.2 $* ...
rsh 11.18.1.2 $*
echo

echo rsh 11.18.1.3 $* ...
rsh 11.18.1.3 $*
echo

echo rsh 11.18.1.4 $* ...
rsh 11.18.1.4 $*
echo

echo rsh 11.18.1.5 $* ...
rsh 11.18.1.5 $*
echo

echo rsh 11.18.1.6 $* ...
rsh 11.18.1.6 $*
echo

echo rsh 11.18.1.7 $* ...
rsh 11.18.1.7 $*
echo

echo rsh 11.18.1.8 $* ...
rsh 11.18.1.8 $*
echo

echo rsh 11.18.1.9 $* ...
rsh 11.18.1.9 $*
echo

echo rsh 11.18.1.10 $* ...
rsh 11.18.1.10 $*
echo

echo rsh 11.18.1.11 $* ...
rsh 11.18.1.11 $*
echo

echo rsh 11.18.1.12 $* ...
rsh 11.18.1.12 $*
echo

echo rsh 11.18.1.13 $* ...
```

```
rsh 11.18.1.13 $*
echo

echo rsh 11.18.1.14 $* ...
rsh 11.18.1.14 $*
echo

echo rsh 11.18.1.15 $* ...
rsh 11.18.1.15 $*
echo

echo rsh 11.18.2.32 $* ...
rsh 11.18.2.32 $*
echo

echo rsh 11.18.2.33 $* ...
rsh 11.18.2.33 $*
echo

echo rsh 11.18.2.34 $* ...
rsh 11.18.2.34 $*
echo

echo rsh 11.18.2.35 $* ...
rsh 11.18.2.35 $*
echo

echo rsh 11.18.2.36 $* ...
rsh 11.18.2.36 $*
echo

echo rsh 11.18.2.37 $* ...
rsh 11.18.2.37 $*
echo

echo rsh 11.18.2.38 $* ...
rsh 11.18.2.38 $*
echo

echo rsh 11.18.2.39 $* ...
rsh 11.18.2.39 $*
echo

echo rsh 11.18.2.40 $* ...
rsh 11.18.2.40 $*
echo

echo rsh 11.18.2.41 $* ...
rsh 11.18.2.41 $*
echo

echo rsh 11.18.2.42 $* ...
rsh 11.18.2.42 $*
echo

echo rsh 11.18.2.43 $* ...
rsh 11.18.2.43 $*
echo

echo rsh 11.18.2.44 $* ...
rsh 11.18.2.44 $*
echo

echo rsh 11.18.2.45 $* ...
rsh 11.18.2.45 $*
echo

echo rsh 11.18.2.46 $* ...
rsh 11.18.2.46 $*
echo

echo rsh 11.18.2.47 $* ...
rsh 11.18.2.47 $*
echo

echo rsh 11.18.3.24 $* ...
```

```
rsh 11.18.3.24 $*
echo

echo rsh 11.18.3.25 $* ...
rsh 11.18.3.25 $*
echo

echo rsh 11.18.3.26 $* ...
rsh 11.18.3.26 $*
echo

echo rsh 11.18.3.27 $* ...
rsh 11.18.3.27 $*
echo

echo rsh 11.18.3.28 $* ...
rsh 11.18.3.28 $*
echo

echo rsh 11.18.3.29 $* ...
rsh 11.18.3.29 $*
echo

echo rsh 11.18.3.30 $* ...
rsh 11.18.3.30 $*
echo

echo rsh 11.18.3.31 $* ...
rsh 11.18.3.31 $*
echo

echo rsh 11.18.1.0 $* ...
rsh 11.18.1.0 $*

echo rsh 11.18.1.1 $* ...
rsh 11.18.1.1 $*

fi
```

B.4 dcfs_mount

```
#!/bin/sh

echo rsh 11.18.1.0 mount /mnt/dcfs...
rsh 11.18.1.0 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.1 mount /mnt/dcfs...
rsh 11.18.1.1 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.2 mount /mnt/dcfs...
rsh 11.18.1.2 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.3 mount /mnt/dcfs...
rsh 11.18.1.3 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.4 mount /mnt/dcfs...
rsh 11.18.1.4 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.5 mount /mnt/dcfs...
rsh 11.18.1.5 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.6 mount /mnt/dcfs...
rsh 11.18.1.6 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.7 mount /mnt/dcfs...
rsh 11.18.1.7 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.8 mount /mnt/dcfs...
rsh 11.18.1.8 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.9 mount /mnt/dcfs...
rsh 11.18.1.9 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.10 mount /mnt/dcfs...
rsh 11.18.1.10 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0
```

```
echo rsh 11.18.1.11 mount /mnt/dcfs...
rsh 11.18.1.11 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.12 mount /mnt/dcfs...
rsh 11.18.1.12 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.13 mount /mnt/dcfs...
rsh 11.18.1.13 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.14 mount /mnt/dcfs...
rsh 11.18.1.14 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.1.15 mount /mnt/dcfs...
rsh 11.18.1.15 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.32 mount /mnt/dcfs...
rsh 11.18.2.32 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.33 mount /mnt/dcfs...
rsh 11.18.2.33 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.34 mount /mnt/dcfs...
rsh 11.18.2.34 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.35 mount /mnt/dcfs...
rsh 11.18.2.35 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.36 mount /mnt/dcfs...
rsh 11.18.2.36 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.37 mount /mnt/dcfs...
rsh 11.18.2.37 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.38 mount /mnt/dcfs...
rsh 11.18.2.38 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.39 mount /mnt/dcfs...
rsh 11.18.2.39 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.40 mount /mnt/dcfs...
rsh 11.18.2.40 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.41 mount /mnt/dcfs...
rsh 11.18.2.41 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.42 mount /mnt/dcfs...
rsh 11.18.2.42 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.43 mount /mnt/dcfs...
rsh 11.18.2.43 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.44 mount /mnt/dcfs...
rsh 11.18.2.44 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.45 mount /mnt/dcfs...
rsh 11.18.2.45 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.46 mount /mnt/dcfs...
rsh 11.18.2.46 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0

echo rsh 11.18.2.47 mount /mnt/dcfs...
rsh 11.18.2.47 mount -t dcfs none /mnt/dcfs -o vol=/dev/dcfs_volumes/volume0
```

B.5 dcfs_lsmount

```
#!/bin/sh

echo rsh 11.18.1.0 "mount | grep dcfs"
rsh 11.18.1.0 mount |grep dcfs

echo rsh 11.18.1.1 "mount | grep dcfs"
rsh 11.18.1.1 mount |grep dcfs

echo rsh 11.18.1.2 "mount | grep dcfs"
rsh 11.18.1.2 mount |grep dcfs
```

```
echo rsh 11.18.1.3 "mount | grep dcfs"
rsh 11.18.1.3 mount |grep dcfs

echo rsh 11.18.1.4 "mount | grep dcfs"
rsh 11.18.1.4 mount |grep dcfs

echo rsh 11.18.1.5 "mount | grep dcfs"
rsh 11.18.1.5 mount |grep dcfs

echo rsh 11.18.1.6 "mount | grep dcfs"
rsh 11.18.1.6 mount |grep dcfs

echo rsh 11.18.1.7 "mount | grep dcfs"
rsh 11.18.1.7 mount |grep dcfs

echo rsh 11.18.1.8 "mount | grep dcfs"
rsh 11.18.1.8 mount |grep dcfs

echo rsh 11.18.1.9 "mount | grep dcfs"
rsh 11.18.1.9 mount |grep dcfs

echo rsh 11.18.1.10 "mount | grep dcfs"
rsh 11.18.1.10 mount |grep dcfs

echo rsh 11.18.1.11 "mount | grep dcfs"
rsh 11.18.1.11 mount |grep dcfs

echo rsh 11.18.1.12 "mount | grep dcfs"
rsh 11.18.1.12 mount |grep dcfs

echo rsh 11.18.1.13 "mount | grep dcfs"
rsh 11.18.1.13 mount |grep dcfs

echo rsh 11.18.1.14 "mount | grep dcfs"
rsh 11.18.1.14 mount |grep dcfs

echo rsh 11.18.1.15 "mount | grep dcfs"
rsh 11.18.1.15 mount |grep dcfs

echo rsh 11.18.2.32 "mount | grep dcfs"
rsh 11.18.2.32 mount |grep dcfs

echo rsh 11.18.2.33 "mount | grep dcfs"
rsh 11.18.2.33 mount |grep dcfs

echo rsh 11.18.2.34 "mount | grep dcfs"
rsh 11.18.2.34 mount |grep dcfs

echo rsh 11.18.2.35 "mount | grep dcfs"
rsh 11.18.2.35 mount |grep dcfs

echo rsh 11.18.2.36 "mount | grep dcfs"
rsh 11.18.2.36 mount |grep dcfs

echo rsh 11.18.2.37 "mount | grep dcfs"
rsh 11.18.2.37 mount |grep dcfs

echo rsh 11.18.2.38 "mount | grep dcfs"
rsh 11.18.2.38 mount |grep dcfs

echo rsh 11.18.2.39 "mount | grep dcfs"
rsh 11.18.2.39 mount |grep dcfs

echo rsh 11.18.2.40 "mount | grep dcfs"
rsh 11.18.2.40 mount |grep dcfs

echo rsh 11.18.2.41 "mount | grep dcfs"
rsh 11.18.2.41 mount |grep dcfs

echo rsh 11.18.2.42 "mount | grep dcfs"
rsh 11.18.2.42 mount |grep dcfs

echo rsh 11.18.2.43 "mount | grep dcfs"
rsh 11.18.2.43 mount |grep dcfs

echo rsh 11.18.2.44 "mount | grep dcfs"
```

```
rsh 11.18.2.44 mount |grep dcfs

echo rsh 11.18.2.45 "mount | grep dcfs"
rsh 11.18.2.45 mount |grep dcfs

echo rsh 11.18.2.46 "mount | grep dcfs"
rsh 11.18.2.46 mount |grep dcfs

echo rsh 11.18.2.47 "mount | grep dcfs"
rsh 11.18.2.47 mount |grep dcfs
```

B.6 dcfs_umount

```
#!/bin/sh

echo "rsh 11.18.1.0 umount /mnt/dcfs*"
rsh 11.18.1.0 'umount /mnt/dcfs*'

echo "rsh 11.18.1.1 umount /mnt/dcfs*"
rsh 11.18.1.1 'umount /mnt/dcfs*'

echo "rsh 11.18.1.2 umount /mnt/dcfs*"
rsh 11.18.1.2 'umount /mnt/dcfs*'

echo "rsh 11.18.1.3 umount /mnt/dcfs*"
rsh 11.18.1.3 'umount /mnt/dcfs*'

echo "rsh 11.18.1.4 umount /mnt/dcfs*"
rsh 11.18.1.4 'umount /mnt/dcfs*'

echo "rsh 11.18.1.5 umount /mnt/dcfs*"
rsh 11.18.1.5 'umount /mnt/dcfs*'

echo "rsh 11.18.1.6 umount /mnt/dcfs*"
rsh 11.18.1.6 'umount /mnt/dcfs*'

echo "rsh 11.18.1.7 umount /mnt/dcfs*"
rsh 11.18.1.7 'umount /mnt/dcfs*'

echo "rsh 11.18.1.8 umount /mnt/dcfs*"
rsh 11.18.1.8 'umount /mnt/dcfs*'

echo "rsh 11.18.1.9 umount /mnt/dcfs*"
rsh 11.18.1.9 'umount /mnt/dcfs*'

echo "rsh 11.18.1.10 umount /mnt/dcfs*"
rsh 11.18.1.10 'umount /mnt/dcfs*'

echo "rsh 11.18.1.11 umount /mnt/dcfs*"
rsh 11.18.1.11 'umount /mnt/dcfs*'

echo "rsh 11.18.1.12 umount /mnt/dcfs*"
rsh 11.18.1.12 'umount /mnt/dcfs*'

echo "rsh 11.18.1.13 umount /mnt/dcfs*"
rsh 11.18.1.13 'umount /mnt/dcfs*'

echo "rsh 11.18.1.14 umount /mnt/dcfs*"
rsh 11.18.1.14 'umount /mnt/dcfs*'

echo "rsh 11.18.1.15 umount /mnt/dcfs*"
rsh 11.18.1.15 'umount /mnt/dcfs*'

echo "rsh 11.18.2.32 umount /mnt/dcfs*"
rsh 11.18.2.32 'umount /mnt/dcfs*'

echo "rsh 11.18.2.33 umount /mnt/dcfs*"
rsh 11.18.2.33 'umount /mnt/dcfs*'

echo "rsh 11.18.2.34 umount /mnt/dcfs*"
rsh 11.18.2.34 'umount /mnt/dcfs*'

echo "rsh 11.18.2.35 umount /mnt/dcfs*"
rsh 11.18.2.35 'umount /mnt/dcfs*'

echo "rsh 11.18.2.36 umount /mnt/dcfs*"
```

```
rsh 11.18.2.36 'umount /mnt/dcfs*'

echo "rsh 11.18.2.37 umount /mnt/dcfs*"
rsh 11.18.2.37 'umount /mnt/dcfs*'

echo "rsh 11.18.2.38 umount /mnt/dcfs*"
rsh 11.18.2.38 'umount /mnt/dcfs*'

echo "rsh 11.18.2.39 umount /mnt/dcfs*"
rsh 11.18.2.39 'umount /mnt/dcfs*'

echo "rsh 11.18.2.40 umount /mnt/dcfs*"
rsh 11.18.2.40 'umount /mnt/dcfs*'

echo "rsh 11.18.2.41 umount /mnt/dcfs*"
rsh 11.18.2.41 'umount /mnt/dcfs*'

echo "rsh 11.18.2.42 umount /mnt/dcfs*"
rsh 11.18.2.42 'umount /mnt/dcfs*'

echo "rsh 11.18.2.43 umount /mnt/dcfs*"
rsh 11.18.2.43 'umount /mnt/dcfs*'

echo "rsh 11.18.2.44 umount /mnt/dcfs*"
rsh 11.18.2.44 'umount /mnt/dcfs*'

echo "rsh 11.18.2.45 umount /mnt/dcfs*"
rsh 11.18.2.45 'umount /mnt/dcfs*'

echo "rsh 11.18.2.46 umount /mnt/dcfs*"
rsh 11.18.2.46 'umount /mnt/dcfs*'

echo "rsh 11.18.2.47 umount /mnt/dcfs*"
rsh 11.18.2.47 'umount /mnt/dcfs*'
```

B.7 dcfs_clean

```
#!/bin/sh

# umounting dcfs ...
./dcfs_umount

# kill clerks
echo rsh 11.18.1.0 killall clerk
rsh 11.18.1.0 killall clerk

echo rsh 11.18.1.1 killall clerk
rsh 11.18.1.1 killall clerk

echo rsh 11.18.1.2 killall clerk
rsh 11.18.1.2 killall clerk

echo rsh 11.18.1.3 killall clerk
rsh 11.18.1.3 killall clerk

echo rsh 11.18.1.4 killall clerk
rsh 11.18.1.4 killall clerk

echo rsh 11.18.1.5 killall clerk
rsh 11.18.1.5 killall clerk

echo rsh 11.18.1.6 killall clerk
rsh 11.18.1.6 killall clerk

echo rsh 11.18.1.7 killall clerk
rsh 11.18.1.7 killall clerk

echo rsh 11.18.1.8 killall clerk
rsh 11.18.1.8 killall clerk

echo rsh 11.18.1.9 killall clerk
rsh 11.18.1.9 killall clerk

echo rsh 11.18.1.10 killall clerk
rsh 11.18.1.10 killall clerk
```

```
echo rsh 11.18.1.11 killall clerk
rsh 11.18.1.11 killall clerk

echo rsh 11.18.1.12 killall clerk
rsh 11.18.1.12 killall clerk

echo rsh 11.18.1.13 killall clerk
rsh 11.18.1.13 killall clerk

echo rsh 11.18.1.14 killall clerk
rsh 11.18.1.14 killall clerk

echo rsh 11.18.1.15 killall clerk
rsh 11.18.1.15 killall clerk

echo rsh 11.18.2.32 killall clerk
rsh 11.18.2.32 killall clerk

echo rsh 11.18.2.33 killall clerk
rsh 11.18.2.33 killall clerk

echo rsh 11.18.2.34 killall clerk
rsh 11.18.2.34 killall clerk

echo rsh 11.18.2.35 killall clerk
rsh 11.18.2.35 killall clerk

echo rsh 11.18.2.36 killall clerk
rsh 11.18.2.36 killall clerk

echo rsh 11.18.2.37 killall clerk
rsh 11.18.2.37 killall clerk

echo rsh 11.18.2.38 killall clerk
rsh 11.18.2.38 killall clerk

echo rsh 11.18.2.39 killall clerk
rsh 11.18.2.39 killall clerk

echo rsh 11.18.2.40 killall clerk
rsh 11.18.2.40 killall clerk

echo rsh 11.18.2.41 killall clerk
rsh 11.18.2.41 killall clerk

echo rsh 11.18.2.42 killall clerk
rsh 11.18.2.42 killall clerk

echo rsh 11.18.2.43 killall clerk
rsh 11.18.2.43 killall clerk

echo rsh 11.18.2.44 killall clerk
rsh 11.18.2.44 killall clerk

echo rsh 11.18.2.45 killall clerk
rsh 11.18.2.45 killall clerk

echo rsh 11.18.2.46 killall clerk
rsh 11.18.2.46 killall clerk

echo rsh 11.18.2.47 killall clerk
rsh 11.18.2.47 killall clerk

#kill servers
echo rsh 11.18.3.24 killall ios
rsh 11.18.3.24 killall ios

echo rsh 11.18.3.25 killall ios
rsh 11.18.3.25 killall ios

echo rsh 11.18.3.26 killall ios
rsh 11.18.3.26 killall ios

echo rsh 11.18.3.27 killall ios
rsh 11.18.3.27 killall ios
```

```
echo rsh 11.18.3.28 killall mgr
rsh 11.18.3.28 killall mgr

echo rsh 11.18.3.29 killall mgr
rsh 11.18.3.29 killall mgr

echo rsh 11.18.3.30 killall mgr
rsh 11.18.3.30 killall mgr

echo rsh 11.18.3.31 killall mgr
rsh 11.18.3.31 killall mgr

#kill cnds
echo rsh 11.18.1.0 killall  cnd scnd
rsh 11.18.1.0 killall cnd scnd
rsh 11.18.1.1 killall scnd scnd

# remove dcfs modules
echo rsh 11.18.1.0 /sbin/rmmod/dcfs
rsh 11.18.1.0 /sbin/rmmod dcfs

echo rsh 11.18.1.1 /sbin/rmmod/dcfs
rsh 11.18.1.1 /sbin/rmmod dcfs

echo rsh 11.18.1.2 /sbin/rmmod/dcfs
rsh 11.18.1.2 /sbin/rmmod dcfs

echo rsh 11.18.1.3 /sbin/rmmod/dcfs
rsh 11.18.1.3 /sbin/rmmod dcfs

echo rsh 11.18.1.4 /sbin/rmmod/dcfs
rsh 11.18.1.4 /sbin/rmmod dcfs

echo rsh 11.18.1.5 /sbin/rmmod/dcfs
rsh 11.18.1.5 /sbin/rmmod dcfs

echo rsh 11.18.1.6 /sbin/rmmod/dcfs
rsh 11.18.1.6 /sbin/rmmod dcfs

echo rsh 11.18.1.7 /sbin/rmmod/dcfs
rsh 11.18.1.7 /sbin/rmmod dcfs

echo rsh 11.18.1.8 /sbin/rmmod/dcfs
rsh 11.18.1.8 /sbin/rmmod dcfs

echo rsh 11.18.1.9 /sbin/rmmod/dcfs
rsh 11.18.1.9 /sbin/rmmod dcfs

echo rsh 11.18.1.10 /sbin/rmmod/dcfs
rsh 11.18.1.10 /sbin/rmmod dcfs

echo rsh 11.18.1.11 /sbin/rmmod/dcfs
rsh 11.18.1.11 /sbin/rmmod dcfs

echo rsh 11.18.1.12 /sbin/rmmod/dcfs
rsh 11.18.1.12 /sbin/rmmod dcfs

echo rsh 11.18.1.13 /sbin/rmmod/dcfs
rsh 11.18.1.13 /sbin/rmmod dcfs

echo rsh 11.18.1.14 /sbin/rmmod/dcfs
rsh 11.18.1.14 /sbin/rmmod dcfs

echo rsh 11.18.1.15 /sbin/rmmod/dcfs
rsh 11.18.1.15 /sbin/rmmod dcfs

echo rsh 11.18.2.32 /sbin/rmmod/dcfs
rsh 11.18.2.32 /sbin/rmmod dcfs

echo rsh 11.18.2.33 /sbin/rmmod/dcfs
rsh 11.18.2.33 /sbin/rmmod dcfs

echo rsh 11.18.2.34 /sbin/rmmod/dcfs
rsh 11.18.2.34 /sbin/rmmod dcfs

echo rsh 11.18.2.35 /sbin/rmmod/dcfs
```

```
rsh 11.18.2.35 /sbin/rmmod dcfs

echo rsh 11.18.2.36 /sbin/rmmod/dcfs
rsh 11.18.2.36 /sbin/rmmod dcfs

echo rsh 11.18.2.37 /sbin/rmmod/dcfs
rsh 11.18.2.37 /sbin/rmmod dcfs

echo rsh 11.18.2.38 /sbin/rmmod/dcfs
rsh 11.18.2.38 /sbin/rmmod dcfs

echo rsh 11.18.2.39 /sbin/rmmod/dcfs
rsh 11.18.2.39 /sbin/rmmod dcfs

echo rsh 11.18.2.40 /sbin/rmmod/dcfs
rsh 11.18.2.40 /sbin/rmmod dcfs

echo rsh 11.18.2.41 /sbin/rmmod/dcfs
rsh 11.18.2.41 /sbin/rmmod dcfs

echo rsh 11.18.2.42 /sbin/rmmod/dcfs
rsh 11.18.2.42 /sbin/rmmod dcfs

echo rsh 11.18.2.43 /sbin/rmmod/dcfs
rsh 11.18.2.43 /sbin/rmmod dcfs

echo rsh 11.18.2.44 /sbin/rmmod/dcfs
rsh 11.18.2.44 /sbin/rmmod dcfs

echo rsh 11.18.2.45 /sbin/rmmod/dcfs
rsh 11.18.2.45 /sbin/rmmod dcfs

echo rsh 11.18.2.46 /sbin/rmmod/dcfs
rsh 11.18.2.46 /sbin/rmmod dcfs

echo rsh 11.18.2.47 /sbin/rmmod/dcfs
rsh 11.18.2.47 /sbin/rmmod dcfs
```

B.8 dcfs_rm

```
#!/bin/sh
# clean the all data and meta data of dcfs

echo rsh 11.18.3.24 ../conf//bin/dcfs_cleanup
rsh 11.18.3.24 ../conf//bin/dcfs_cleanup

echo rsh 11.18.3.24 rm ../conf//conf/l*/l*
rsh 11.18.3.24 rm -f ../conf//conf/l*/l*

echo rsh 11.18.3.25 ../conf//bin/dcfs_cleanup
rsh 11.18.3.25 ../conf//bin/dcfs_cleanup

echo rsh 11.18.3.25 rm ../conf//conf/l*/l*
rsh 11.18.3.25 rm -f ../conf//conf/l*/l*

echo rsh 11.18.3.26 ../conf//bin/dcfs_cleanup
rsh 11.18.3.26 ../conf//bin/dcfs_cleanup

echo rsh 11.18.3.26 rm ../conf//conf/l*/l*
rsh 11.18.3.26 rm -f ../conf//conf/l*/l*

echo rsh 11.18.3.27 ../conf//bin/dcfs_cleanup
rsh 11.18.3.27 ../conf//bin/dcfs_cleanup

echo rsh 11.18.3.27 rm ../conf//conf/l*/l*
rsh 11.18.3.27 rm -f ../conf//conf/l*/l*

echo rsh 11.18.3.28 ../conf//bin/dcfs_cleanup
rsh 11.18.3.28 ../conf//bin/dcfs_cleanup

echo rsh 11.18.3.28 rm ../conf//conf/l*/l*
rsh 11.18.3.28 rm -f ../conf//conf/l*/l*

echo rsh 11.18.3.29 ../conf//bin/dcfs_cleanup
rsh 11.18.3.29 ../conf//bin/dcfs_cleanup
```

```
echo rsh 11.18.3.29 rm ../conf//conf/l*/l*
rsh 11.18.3.29 rm -f ../conf//conf/l*/l*

echo rsh 11.18.3.30 ../conf//bin/dcfs_cleanup
rsh 11.18.3.30 ../conf//bin/dcfs_cleanup

echo rsh 11.18.3.30 rm ../conf//conf/l*/l*
rsh 11.18.3.30 rm -f ../conf//conf/l*/l*

echo rsh 11.18.3.31 ../conf//bin/dcfs_cleanup
rsh 11.18.3.31 ../conf//bin/dcfs_cleanup

echo rsh 11.18.3.31 rm ../conf//conf/l*/l*
rsh 11.18.3.31 rm -f ../conf//conf/l*/l*

echo rsh 11.18.1.0 ../conf//bin/dcfs_cleanup
rsh 11.18.1.0 ../conf//bin/dcfs_cleanup

echo rsh 11.18.1.0 rm ../conf//conf/l*/l*
rsh 11.18.1.0 rm -f ../conf//conf/l*/l*

echo rsh 11.18.1.1 ../conf//bin/dcfs_cleanup
rsh 11.18.1.1 ../conf//bin/dcfs_cleanup

echo rsh 11.18.1.1 rm ../conf//conf/l*/l*
rsh 11.18.1.1 rm -f ../conf//conf/l*/l*
```

B.9 dcfs_cleanup

```
#!/bin/sh

META_ROOT="/home/wsn/meta"
DATA_ROOT="/home/wsn/data"

echo "clear meta data..."
cd $META_ROOT
pwd
PWD=`pwd`

if [ "$PWD" = "$META_ROOT" ]; then
     VOLUMES=`ls`
     echo $VOLUMES
     for v in $VOLUMES; do
          echo rm -f -r $v
          rm -f -r $v
     done
fi

####################

echo "clear file data ..."
cd $DATA_ROOT
pwd
PWD=`pwd`

if [ "$PWD" = "$DATA_ROOT" ]; then
     VOLUMES=`ls`
     echo $VOLUMES
     for v in $VOLUMES; do
          echo rm -f -r $v
          rm -f -r $v
     done
fi

echo "----------- data files --------------------"
echo ls -l $DATA_ROOT
ls -l $DATA_ROOT
echo "--------- end data files ------------------"


echo "========== meta files ======================"
echo "ls -a $META_ROOT"
ls -a $META_ROOT
echo "======== end meta files ====================="
```

附录 C DCFS Source

```
broker/
broker/include
broker/include/b2c_const.h
broker/include/b2c_request.h
broker/include/b2c_share.h
broker/include/broker.h
broker/include/broker_extern.h
broker/include/broker_perf.h
broker/include/broker_rw.h
broker/include/dcfs_dir.h
broker/include/dcfs_ioctl.h
broker/include/dcfs_ksys.h
broker/source
broker/source/Makefile
broker/source/b2c_request.c
broker/source/dcfs_init.c
broker/source/dcfs_mmap.c
broker/source/dcfs_mod.c
broker/source/dcfs_shm.c
broker/source/dir.c
broker/source/file.c
broker/source/low_level.c
broker/source/super.c
broker/source/symlink.c
clerk/
clerk/include
clerk/include/clerk_comm.h
clerk/include/clerk.h
clerk/include/clerk_errno.h
clerk/include/clerk_extern.h
clerk/include/clerk_perf.h
clerk/source
clerk/source/Makefile
clerk/source/broker_req.c
clerk/source/c2m_request.c
clerk/source/clerk.c
clerk/source/clerk_debug.c
clerk/source/clk
clerk/source/comm_broker.c
clerk/source/comm_cnd.c
clerk/source/comm_server.c
clerk/source/server_ack.c
clerk/source/wait_queue.c
COM/
COM/include
COM/include/DList.h
COM/include/__com_bcl.h
COM/include/__com_ethpub.h
COM/include/__com_ethpub.h.bak
COM/include/__com_jaywalker.h
COM/include/__com_msg.h
```

```
COM/include/__com_port.h
COM/include/__com_tcp.h
COM/include/__com_udp.h
COM/include/__com_wait.h
COM/include/com.h
COM/include/usr_types.h
COM/README
COM/lib
COM/source
COM/source/Makefile
COM/source/README
COM/source/bcl.c
COM/source/com.c
COM/source/ethpub.c
COM/source/jaywalker.c
COM/source/tcp.c
COM/source/udp.c
conductor/
conductor/adm-utils
conductor/adm-utils/include
conductor/adm-utils/include/__adm_basicfg.h
conductor/adm-utils/include/__adm_clkcfg.h
conductor/adm-utils/include/__adm_common.h
conductor/adm-utils/include/__adm_const.h
conductor/adm-utils/include/__adm_ethpub.h
conductor/adm-utils/include/__adm_lv.h
conductor/adm-utils/include/__adm_mount.h
conductor/adm-utils/include/__adm_svrcfg.h
conductor/adm-utils/include/adm-util.h
conductor/adm-utils/source
conductor/adm-utils/source/Makefile
conductor/adm-utils/source/README
conductor/adm-utils/source/__adm_clerkadd.c
conductor/adm-utils/source/__adm_clerkdown.c
conductor/adm-utils/source/__adm_clerkup.c
conductor/adm-utils/source/__adm_cndown.c
conductor/adm-utils/source/__adm_ethpub.c
conductor/adm-utils/source/__adm_lvcrt.c
conductor/adm-utils/source/__adm_lvmnt.c
conductor/adm-utils/source/__adm_lvumnt.c
conductor/adm-utils/source/__adm_macro.c
conductor/adm-utils/source/__adm_mcndup.c
conductor/adm-utils/source/__adm_scndup.c
conductor/adm-utils/source/__adm_svradd.c
conductor/adm-utils/source/__adm_svrdown.c
conductor/adm-utils/source/__adm_svrup.c
conductor/adm-utils/source/__adm_sysinfo.c
conductor/adm-utils/source/adm-util.c
conductor/adm-utils/source/adm-util
conductor/adm-utils/source/ncic.gif
conductor/adm-utils/source/__adm_basicfg.c
conductor/include
conductor/include/__cnd_HA.h
```

```
conductor/include/__cnd_adm.h
conductor/include/__cnd_clerkadd.h
conductor/include/__cnd_clerkup.h
conductor/include/__cnd_debug.h
conductor/include/__cnd_fsinfo.h
conductor/include/__cnd_hb.h
conductor/include/__cnd_help.h
conductor/include/__cnd_lvcrt.h
conductor/include/__cnd_mesg.h
conductor/include/__cnd_mloop.h
conductor/include/__cnd_objdown.h
conductor/include/__cnd_pgrp.h
conductor/include/__cnd_port.h
conductor/include/__cnd_scnd.h
conductor/include/__cnd_svradd.h
conductor/include/__cnd_svrinfo.h
conductor/include/__cnd_svrup.h
conductor/include/cnd.h
conductor/source
conductor/source/Makefile
conductor/source/README
conductor/source/__cnd_HA.c
conductor/source/__cnd_clerkadd.c
conductor/source/__cnd_clerkup.c
conductor/source/__cnd_hb.c
conductor/source/__cnd_help.c
conductor/source/__cnd_lvcrt.c
conductor/source/__cnd_objdown.c
conductor/source/__cnd_scnd.c
conductor/source/__cnd_svradd.c
conductor/source/__cnd_svrup.c
conductor/source/cnd.c
FSCK
FSCK/comm.c
FSCK/comm.h
FSCK/error.c
FSCK/error.h
FSCK/fsck.c
FSCK/fsck.h
FSCK/Makefile
FSCK/pass1.c
FSCK/pass2.c
FSCK/pass3.c
FSCK/pass4.c
FSCK/pass5.c
FSCK/pass6.c
FSCK/dcfs_mgr_fsck.c
help/
help/include
help/include/thread.h
help/include/help.h
help/lib
help/source
```

```
help/source/Makefile
help/source/help.c
help/source/thread.c
include/
include/README
include/const.h
include/msgtype.h
include/debug.h
include/fs.h
include/msg.h
include/thread.h
include/sys.h
include/types.h
include/list
include/list/hash.c
include/list/hash.h
include/list/list.h
include/list/slist.h
include/list/tailq.h
IOS/
IOS/include
IOS/include/ios_cache.h
IOS/include/ios.h
IOS/include/ios_cnd.h
IOS/include/ios_exec_threads.h
IOS/include/ios_flush.h
IOS/include/ios_help.h
IOS/include/ios_ops.h
IOS/include/ios_rdwr.h
IOS/include/ios_reqs.h
IOS/include/ios_scheduler.h
IOS/source
IOS/source/Makefile
IOS/source/ios.c
IOS/source/ios_cache.c
IOS/source/ios_cnd.c
IOS/source/ios_scheduler.c
IOS/source/ios_flush.c
IOS/source/ios_help.c
IOS/source/ios_ops.c
IOS/source/ios_rdwr.c
IOS/source/ios_exec_threads.c
IOS/test
IOS/test/Makefile
MGR/
MGR/include
MGR/include/mgr.h
MGR/include/mgr_cnd.h
MGR/include/mgr_dir.h
MGR/include/mgr_flock.h
MGR/include/mgr_flush.h
MGR/include/mgr_fs.h
MGR/include/mgr_help.h
```

```
MGR/include/mgr_inode.h
MGR/include/mgr_objs.h
MGR/include/mgr_super.h
MGR/include/mgr_undo.h
MGR/source
MGR/source/Makefile
MGR/source/mgr.c
MGR/source/mgr_cnd.c
MGR/source/mgr_dir.c
MGR/source/mgr_flock.c
MGR/source/mgr_flush.c
MGR/source/mgr_help.c
MGR/source/mgr_inode.c
MGR/source/mgr_objs.c
MGR/source/mgr_super.c
MGR/source/mgr_undo.c
MGR/test
MGR/test/Makefile
tools/
tools/dcfs_cfg.c
tools/Makefile
tools/clk
tools/dcfs_cleanup
Makefile
Makefile.conf
COPYRIGHT
```