Open Software Foundation
Request For Comments: 88.0
December 1995

J. Gait (Transarc)
A. Khale (Transarc)
J. Morin (Transarc)

A JUKEBOX BACKUP SUBSYSTEM FOR DFS

1. INTRODUCTION

The current DFS backup subsystem requires a backup operator to respond to queries from the device controller. In particular **storage units** (tapes) have to be manually loaded. Modern stacker and jukebox equipment encourage more flexible approaches to managing backups that focus on storage unit autoloading and default responses. This RFC describes a methodology for gracefully integrating stackers and jukeboxes into the DFS backup subsystem in a way that supports unattended backup of large-scale DFS file systems.

2. OVERVIEW OF DFS BACKUP

The DFS backup subsystem supports dump and restore of DFS filesets to and from tape. (The restoration of an individual file entails the restoration of its containing fileset.) There are four areas of strength in the DFS backup subsystem:

(a) *On-line backup.* The DFS backup subsystem supports on-line dump of DFS filesets during operations with minimal impact. (The mechanism is to dump clones of the target filesets.)

(b) *Single point of control.* Since it is feasible to carry on simultaneous backup to multiple **storage devices** (tape drives) on multiple machines in the cell, it is possible for a great deal of management complexity to be introduced. The backup subsystem is designed to manage complex activities as gracefully as possible by integrating all backup operator actions at a single node in the DFS cell.

(c) *Multi-fileset atomicity.* It is possible to specify a family of filesets that is atomically dumped and restored. This feature supports coherent backup and restore of mutually interdependent filesets. (Most commonly this feature is used to treat the filesets in an aggregate together for purposes of backup.)

(d) *Backup for moving filesets.* The files stored in DFS may be transparently moved among the file servers in the cell at (almost) any time. Thus the backup subsystem is designed to handle moving filesets.

Each storage unit is **labeled** by the backup subsystem. The label is written into the first 16 kbytes and contains a name for the storage unit, a specification of the fileset being dumped, a dump ID and date, and an expiration date, among other things. The label is written in an internal DFS format — currently there is no support for ANSI standard or IBM standard labels. DFS does not currently support permanent names for storage units. (A permanent name could be known to the jukebox and used to locate a particular storage unit during a restore.)

There are three components to the backup subsystem: the **device coordinator**, the **backup program**, and the **backup database**. These are discussed in the following subsections.

2.1. Device Coordinator

Each storage device used by the DFS backup subsystem is attached to a DFS server machine that runs a process called a **device coordinator**, or **backup tape coordinator** (`butc`), for that storage device. The device coordinator is specific to the storage device being coordinated, so if there are multiple storage devices attached to the DFS server, then there is a device coordinator for each one. The device coordinator is responsible for managing the storage device, its **device configuration file** and log files stored on the same machine. The set of device coordinators in a DFS cell are distinguished from one another by cell-wide unique device coordinator IDs.

A device coordinator stores each fileset contiguously on a storage unit, or on multiple storage units if necessary, and separates filesets by an EOF, or **end of fileset** mark. The size of the EOF is storage device dependent and may be as large as 2 mbytes. The device configuration file for the storage device contains the name of the device, the size of the storage unit, the size of the EOF mark, and the device controller ID.

The device coordinator has interactions with five other entities:

(a) With the storage device while managing the storage units.

(b) With the DFS file exporter while dump and restore operations are in process.

(c) With the user of the backup subsystem to effect physical operations that require stereotypical operator action.

(d) With the local file system to configure the storage device and to log operations.

(e) With the cell backup database.

2.1.1. Interaction with the operator

There are a number of explicit interactions between the device coordinator and the operator:

(a) The operator is prompted for the first storage unit.

(b) The operator is prompted to change the storage unit.

(c) When a restore operation experiences a failure, the operator is prompted whether to proceed.

(d) When a dump fileset operation fails, the operator is prompted whether to retry the fileset, omit the fileset or abort the operation.

(e) The operator may be prompted during `scantape` operations whether there is an additional storage unit to be scanned.

(f) The operator may be prompted whether to label a storage unit.

Any of these events halts backup operations subject to the intervention of the backup operator. Prompting for the first storage unit can be turned off by starting the device coordinator with the `noautoquery` flag. In this case the first storage unit is assumed to be loaded.

2.2. Backup Program

This is the central point for managing the backup subsystem and is run by the backup operator on any client machine in the DFS cell. This **backup program** (`bak`) has four points of interaction with other entities:

(a) With the operator. Tape coordinator operator interactions are brought to the same client machine, but different windows.

(b) With the ensemble of device coordinators hosted on various other server machines in the cell.

(c) With the cell backup database. (A log is also kept and stored locally.)

(d) With the DFS FLDB to obtain the locations of filesets.

The backup program is the focus for the definition of fileset families.

2.3. Backup Database

The cell-wide **backup database** records dump schedules, locations of device coordinators, fileset families, etc. The backup database is maintained by a DFS server called the `bakserver`. The backup database must be complete and consistent to guarantee that filesets can be restored, so it must itself be backed-up. Ultimately the backup of the backup database resolves to halting DFS activity on the host, usually via `bos` commands, and a resort to host vendor provided backup utilities or to specialized DFS backup subsystem commands.

3. JUKEBOX BACKUP

Stackers and jukeboxes are capable of switching between physical storage units automatically during dump operations, and jukeboxes are capable of automatically switching between the correct physical storage units during restore. The DFS backup subsystem supports these devices by providing an **auto operation configuration file** that is read by `butc` when it starts a device coordinator process for the device, configuring the device coordinator with the information taken from the auto operation configuration file. The auto operation configuration file is placed in the directory `dcelocal/var/dfs/backup`, and is given a name of the form `conf_`*`device`*, where `device` specifies a particular jukebox or stacker. Currently DFS uses the path name of the jukebox or stacker to specify `device`. (Note: the *auto* operation configuration file is different from the ordinary device configuration file, which defines the physical device to the device coordinator.)

The auto operation configuration file supports these parameters:

(a) `MOUNT`: The path to a procedure that performs mount operations on physical storage units.

(b) `UNMOUNT`: The path to a procedure that performs unmount operations on physical storage units.

(c) `ASK`: Configures a set of default answers to prompts, except for the initial prompt from the device coordinator.

(d) `NAME_CHECK`: Normally the names of physical storage units are checked; this parameter enables bypassing the name check, hence makes it possible to recycle storage units without relabeling them.

(e) `FILE`: It is possible to use this mechanism to dump and restore to and from a set of files, rather than to and from a set of storage units in a stacker or jukebox; this parameter relates to this capability.

The `ASK` parameter is used in conjunction with the `noautoquery` startup flag to `butc` to automate, as nearly as is possible, the manipulation of physical storage units. Any error while operating on storage devices is interpreted by the device coordinator in a manner that is as compatible as possible with unattended operation, as follows:

(a) `labeltape`: If an error occurs during a `labeltape` operation, then the device coordinator prompts for another tape.

(b) `dump`: An error during a `dump` operation indicates a fileset error and the device coordinator accordingly omits the fileset.

(c) `restore`: An error during `restore` indicates a fileset error, so the device coordinator omits the fileset.

(d) `scantape`: An error during a `scantape` operation means there is a problem in determining the next storage device; the device coordinator prompts for it.

These parameters are passed to the `mount` procedure:

(a) The path to the device.

(b) The operation to be performed.

(c) The number of times the physical storage unit has been requested.

(d) The name of the storage unit.

(e) The dump identifier.

The operation to be performed is drawn from one of the `bak` commands: `dump`, `labeltape`, `readlabel`, `restore`, and `scantape`.

These parameters are passed to the `unmount` procedure:

(a) The path to the device.

(b) The operation to be performed.

For the `unmount` procedure the operation to be performed is always the `bak` command `unmount`.

These scripts are written by the system administrator and are executed by the device coordinator. The system administrator must still verify that filesets were successfully dumped and must still physically update the external label on the storage unit that contains the dumped fileset.

4. CHANGES TO BACKUP FOR JUKEBOXES

The changes required in the DFS backup subsystem to support stackers and jukeboxes are pervasive but not particularly extensive. The two important things to do are to make all interactions between the device coordinator and the backup operator selective and settable from the auto operation configuration file, and to supply a callout routine that invokes the procedures for `mount` and `unmount` that are referenced in the auto operation configuration file and defined by the system administrator on a per-device basis.

4.1. Configuring the Tape Coordinator

Here are the four configurable parameters that have been introduced, together with their default values:

```
dump_namecheck = 1;
queryoperator  = 1;
opencallout    = (char *)0;
closecallout   = (char *)0;
```

So by default storage unit names are checked, operations are interactive and there is no callout to external routines to mount and unmount storage units. These values are resettable by the device coordinator at startup to the values found in the auto operation configuration file. For a stacker or jukebox the system administrator would set `queryoperator` to 0 so operation would be non-interactive, and would supply values for `opencallout` and `closecallout` that would automatically mount and unmount storage units in the device.

4.2. Callout to an External Procedure

Here is the callout pseudocode that supports automatic mount and unmount:

```
if (fork() == 0)
{
    callOut = opencallout;
    switch (opflag)
    {
        case READOPCODE:
            strcpy(opcode, "restore");
            break;
        case WRITEOPCODE:
            strcpy(opcode, "dump");
            break;
        case LABELOPCODE:
            strcpy(opcode, "labeltape");
            break;
        case READLABELOPCODE:
            strcpy(opcode, "readlabel");
            break;
        case SCANOPCODE:
            strcpy(opcode, "scantape");
            break;
        case CLOSEOPCODE:
            strcpy(opcode, "unmount");
            callOut = closecallout;
            break;
    }
    CO_argv[0] = callOut;
    CO_argv[1] = tapePath;
    CO_argv[2] = opcode;
    if (opflag == CLOSEOPCODE)
        CO_argv[3] = (char *)0;
    else
    {
        CO_argv[3] = tapecount;
        if (name)
            CO_argv[4] = name;
        if (dbDumpId)
```

```
                CO_argv[5] = dbDumpId;
            CO_argv[6] = (char *)0;
        }
        code = execve(callOut, CO_argv, (char *)0);
    }
```

The callout routine is passed a flag that indicates the opcode, and this information is combined with global values that describe the state of the storage device and the backup activity to construct a callout to the `mount` or `unmount` routine provided by the system administrator. The callout routine is invoked whenever the device coordinator performs one of the indicated activities and the default values of the configuration parameters have been reset for unattended operation.

The callout routine passes five arguments (`CO_argv[1]` through `CO_argv[5]`) to the `mount` routine:

(a) The pathname of the storage device.

(b) The operation to be performed.

(c) The number of times the device coordinator has called the script while repeatedly attempting this same operation. (For example, in case there is some physical problem with the storage unit.)

(d) The name of the storage unit.

(e) The tape ID.

Only the first two of these arguments are passed to the `unmount` routine.

5. EXAMPLE: JUKEBOX EMULATION

The `FILE` parameter described above allows stacker and jukebox operation to be emulated using a set of files in place of the storage units of a multi-unit storage device. This feature separates the behaviors of the software backup system from the behaviors of physical devices and dramatically improves the quality of life for developers of backup code. This presentation serves to make the new technology concrete in a way that does not depend on specific stacker or jukebox implementation.

Here is a sample auto operation configuration file that might be used to emulate the physical storage device used for backup using a set of files:

```
MOUNT /opt/dcelocal/var/dfs/backup/DEVICE/mount
UNMOUNT /opt/dcelocal/var/dfs/backup/DEVICE/unmount
ASK NO
NAME_CHECK YES
FILE YES
```

This file is read by `butc` when the backup subsystem is started. The auto operation configuration file specifies executable procedures for `mount` and `unmount`, disables all interaction with the backup operator, enables name checking and directs that a file is to be used instead of a physical storage unit. The path to the file is specified in the device configuration file.

Here is a (simplified) shell procedure for the `MOUNT` parameter:

```
Path=$1
Operation=$2
Requests=$3
TapeName=$4
TapeControllerId=$5
DATA_DIR=`dirname $Path`/data
SCAN_LOCK=$DATA_DIR/.scan
if [ $Operation = "scantape" ]
then
    ROOT=`echo $Path | \
        awk -F"." '{ORS=".";for(i=1;i<NF;++i)print $i}'`
```

```
        EXT=`echo $Path | \
            awk -F"." '{print $NF}'`
        if [ -f $SCAN_LOCK ]
        then
            FILE="$ROOT`expr $EXT + 1`"
        else
            touch $SCAN_LOCK
            FILE="$ROOT1"
        fi
        if [ -f $FILE ]
        then
            rm -f $Path
            ln -s $FILE $Path
            exit 0
        else
            exit 1
        fi
    fi
    if [ $Operation = "labeltape" ]
    then
        exit 0
    fi
    if [ ($Operation != "dump") -a ($Operation != "restore") ]
    then
        exit 1
    fi
    rm -f $SCAN_LOCK
    if [ -f $Path ]
    then
        rm -f $Path
    fi
    if [ ! -d $DATA_DIR ]
    then
        mkdir -p $DATA_DIR
    fi
    if [ ! -f $DATA_DIR/$TapeControllerId.$TapeName ]
    then
        touch $DATA_DIR/$TapeControllerId.$TapeName
    fi
    ln -s $DATA_DIR/$TapeControllerId.$TapeName $Path
    exit $?
```

Notice that the `Path` parameter is actually a link to a file. The link represents the currently mounted physical storage unit. The `Requests` parameter is not used in this script but would normally be used in scripts to prevent infinite loops. An error code of 1 from the script instructs the device coordinator to end the operation, and any error code greater than 1 directs the device coordinator to prompt the operator.

Here is a shell procedure for the `UNMOUNT` parameter:

```
Path=$1
Operation=$2
if [ -f $Path ]
then
    rm -f $Path
fi
exit 0
```

Here is the content of a sample device configuration file designed for emulating physical storage with a set of files:

```
140M 200K /opt/dcelocal/var/dfs/backup/DEVICE/link 0
```

The value of `link` is a symbolic link to the file that emulates the currently mounted storage unit. This storage unit stores 140 mbytes, uses 200 kbyte EOF marks, and is controlled by device controller 0. (Actually, when dumping to files the device coordinator does not write EOF marks, so the 200 kbyte value is ignored in the above example.)

6. ACKNOWLEDGEMENTS

The following persons made significant contributions to the stacker and jukebox backup subsystem for DFS:

(a) Rick Welch (Naval Research Laboratory).

(b) Peter Chan (IBM).

(c) Jayant Ramakrishnan (University of Maryland).

AUTHORS' ADDRESSES

Jason Gait
Transarc Corp.
707 Grant Street
Pittsburgh, PA 15219
USA
Internet email: gait@transarc.com
Telephone: +1-412-338-6933

Abhijit Khale
Transarc Corp.
707 Grant Street
Pittsburgh, PA 15219
USA
Internet email: khale@transarc.com
Telephone: +1-412-338-6956

John Morin
Transarc Corp.
707 Grant Street
Pittsburgh, PA 15219
USA
Internet email: morin@transarc.com
Telephone: +1-412-338-6908