AdvFS on-disk structures

Module 2

Objectives

To explain AdvFS on-disk structures, you should be able to describe:

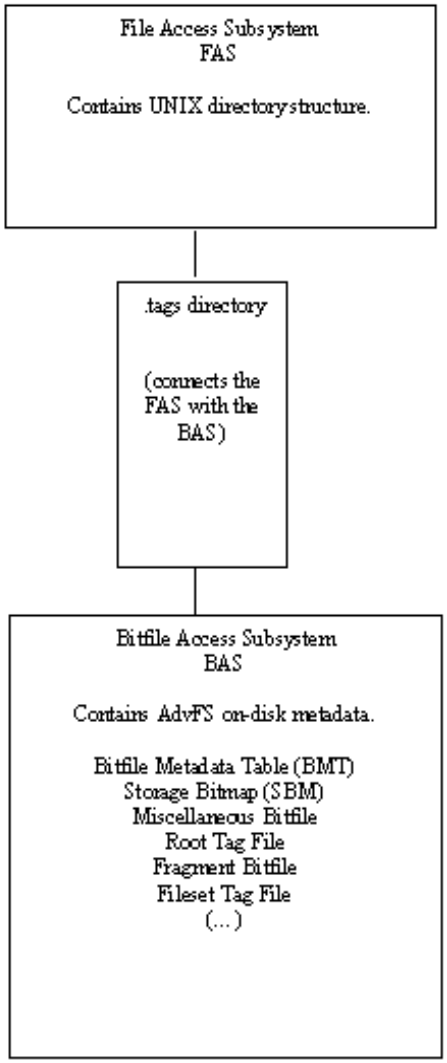
- Bitfiles
- Mcells
- Extent maps
- Tags
- Directory files and directory index files
- Fragments
- Storage bitmap bitfiles
- Quota files

Two-level implementation of AdvFS (1 of 2)

- File Access System (FAS)
  - the higher level of AdvFS
  - transforms bitfiles into normal UNIX files
  - the client layer
- Bitfile Access System (BAS)
  - the lowest level of AdvFS providing storage support

Two-level implementation of AdvFS (2 of 2)

On Disk Structures



.tags directory

- Each AdvFS file system has a .tags directory
- Allows files to be accessed by tag, sequence number
  - `/Advfs_mount_point/.tags/15374`
  - `/Advfs_mount_point/.tags/0x3c0e.8001`

Tag number can access any file

```
# ls -li
total 31
22894 -rwxr-xr-x  1 root system 31114 Jun 24 15:20
ob_1
#
# tail -3 ob_1
:of#169728:pf#35135:bf#8192:ff#1024:\
:og#99458:pg#149368:bg#8192:fg#1024:\
:oh#0:ph#0:bh#8192:fh#1024:
#
# tail -3 /usr/.tags/22894
:of#169728:pf#35135:bf#8192:ff#1024:\
:og#99458:pg#149368:bg#8192:fg#1024:\
:oh#0:ph#0:bh#8192:fh#1024:
#
```

BAS on-disk format: Everything is a bitfile

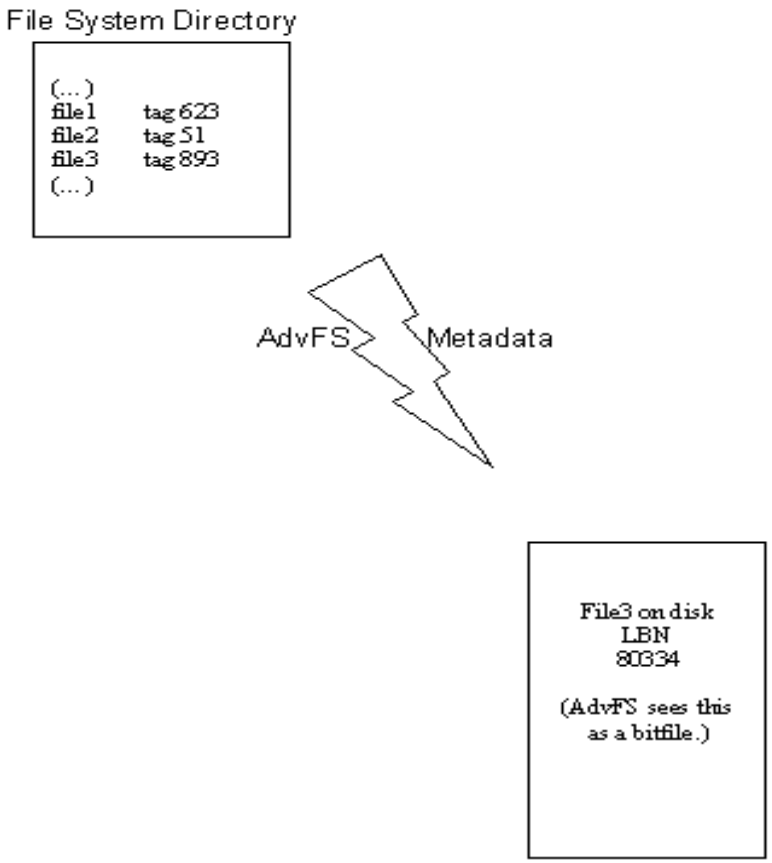
- Bitfiles are arrays of 8K disk pages holding user data or metadata
- A series of contiguous 8K pages in a bitfile is stored as an extent
- Each bitfile is identified by its tag
  - Tag consists of a tag number.sequence number pair
- Tag number can be used to locate the extents of a file

Note that `tag2name` has a new format for V5.

Tag number 22894 being translated by tag2name

```
# /sbin/advfs/tag2name /usr/.tags/22894
/usr/bruden/ob_1
#
# echo $PATH
/sbin:/usr/sbin:/usr/bin:/usr/ccs/bin:/usr/bin/X11:/usr
/local
#
# PATH=$PATH:/sbin/advfs
#
# tag2name usr_domain -S usr 22894
open_vol: open for volume "/dev/disk/dsk2g" failed:
Device busy
#
# tag2name -r usr_domain -S usr 22894     <- Uses raw
device (-r)
bruden/ob_1
```

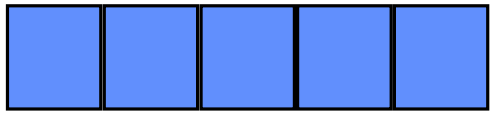
Using AdvFS metadata to translate FAS to BAS



BAS on-disk format

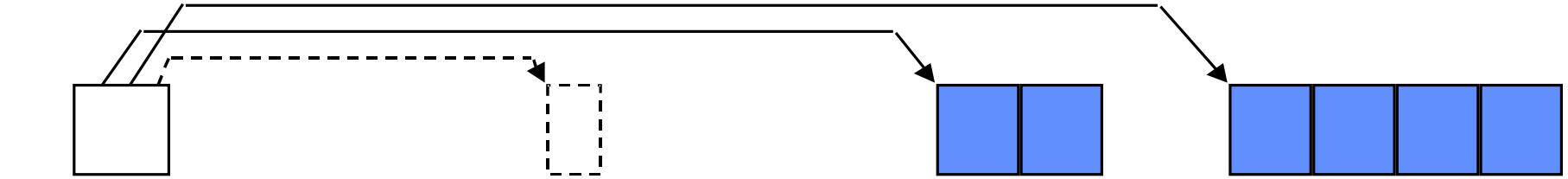
Logical File

owner
group
size
mod bits
....



8K Pages

On Disk



(Primary) mcell
292 bytes
Contains variable sized records such as;
POSIX attributes
extent map records

Additional mcell(s)
optional
can contain more extent map records if needed

extent 1

extent 2

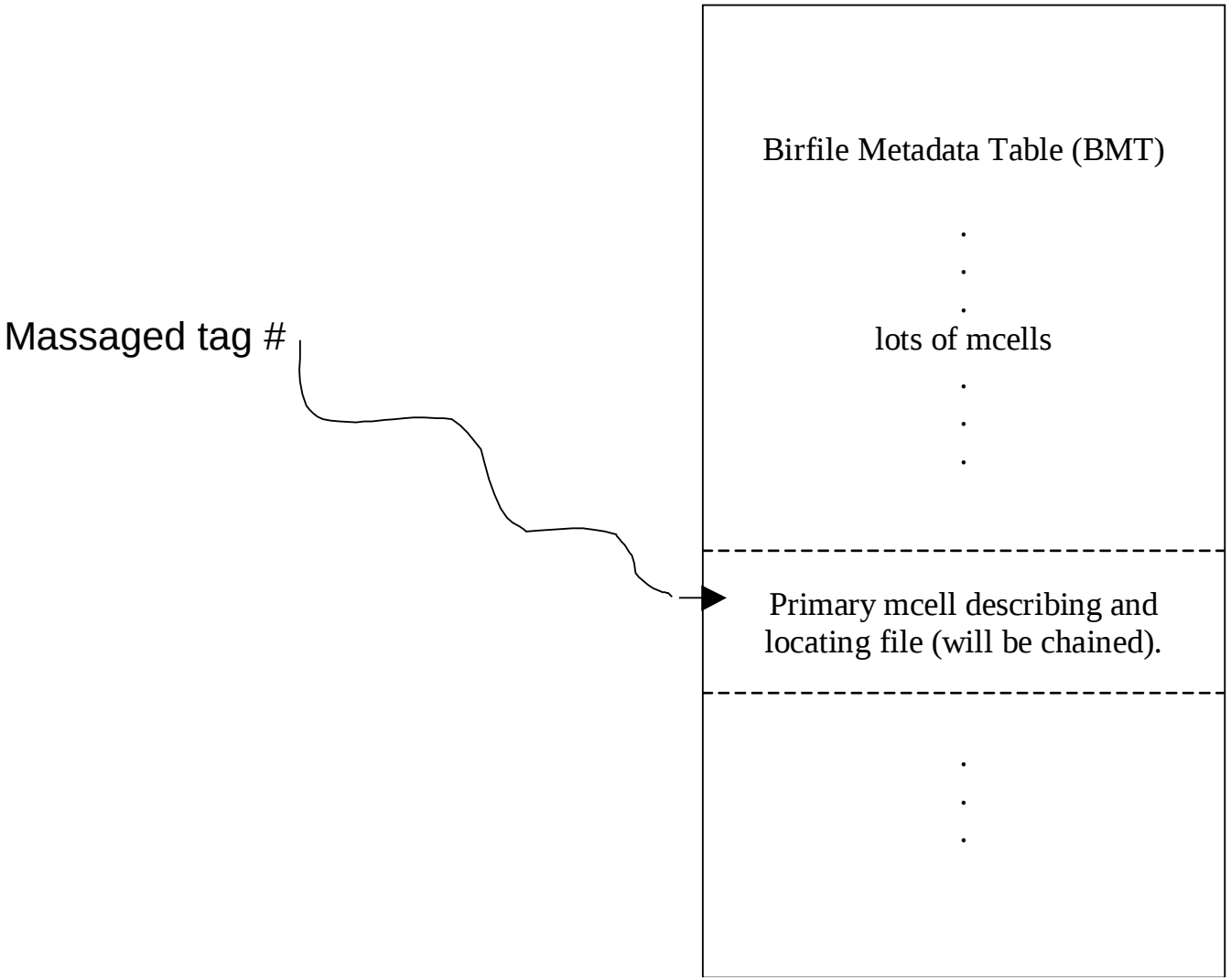
Bitfiles

- Array of 8K pages
- Stored as extents
  - groups of on-disk contiguous 8K pages
  - managed by extent maps
- Identified by a tag
  - tag.sequence, for example 4714.8001
  - tag number is similar to an inode number
  - sequence number functions as a generation number
- All sectors are free or in a bitfile
- Managed by mcell chains

Mcells

- AdvFS locates extents by finding the file's Primary Metadata cell (Mcell) and stepping through the mcells to find the extent information
  - tag number is like an inode number, but an mcell functions like an inode
  - holds permissions, size, extent info, link count, and so forth
- Each mcell is 292 bytes
  - can fit 28 on an 8K page (plus a 16-byte header)
- Where are the mcells located?

Tag number to BMT mcell to logical blocks.



AdvFS file addresses

AdvFS files have a means of identification:

- Tag
  - very similar to the UFS inode number
  - can be discovered with the ls -i command
- Primary mcell ID
  - component of the lower BAS layer
  - start of a linked list of one or more mcells
  - well hidden from users and system administration

Bitfile-sets

- FAS fileset represents a BAS bitfile-set
- Identified by numbers
- Bitfiles are known by:
  - domain ID
  - fileset ID
  - tag, sequence number

Reusing tags

- Tag numbers can be reused
  - with file creation and deletion
  - like inode numbers
- Sequence number identifies various lives of the tag
  - tags have initial sequence number of 8001 hexadecimal or 32769 decimal
  - sequence number is incremented when tag number is reused
  - when sequence number overflows, tag is discarded
  - leftmost bit indicates ‘tag in use’, remaining 15 bits are used for sequencing
    - hexadecimal 8001 is binary 1000000000000001

Domain and volume structures

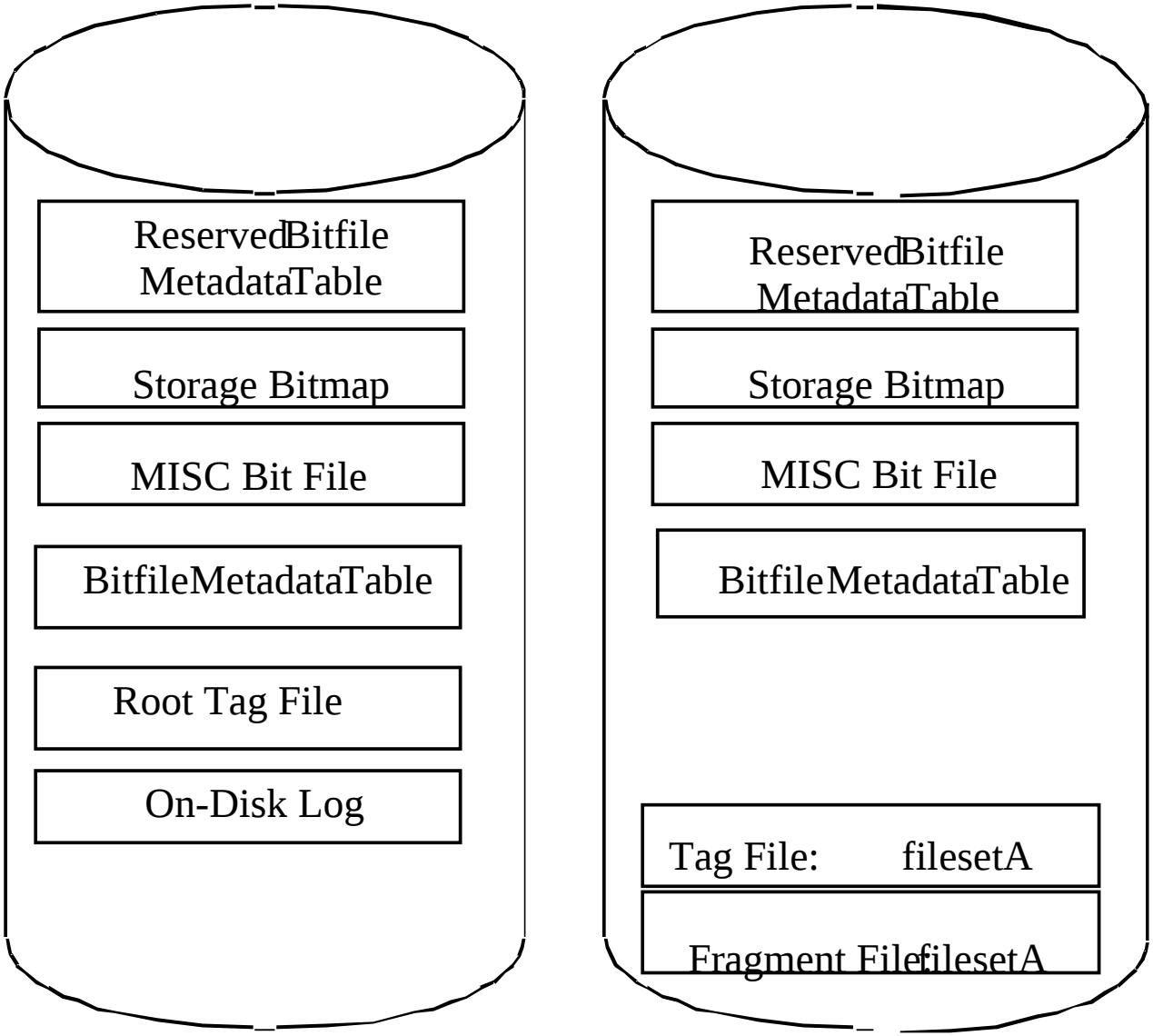
- Reserved bitfile metadata table
- Bitfile metadata table
- Storage bitmap
- Misc bitfile
- Transaction log
- Root tag file

BAS on-disk metadata bitfiles

per volume

per domain

per fileset



Per domain bitfiles

- On-disk log
  - contains the transaction log
  - usually 4MB in size
- Root tag file
  - lists filesets
  - one page in size

Displaying the root tag file

```
# nvtagpg -r usr_domain


================================================================================
DOMAIN "usr_domain"  VDI 1 (/dev/rdisk/dsk2g)  lbn 96   root TAG page 0
--------------------------------------------------------------------------------
currPage 0
numAllocTMaps 3  numDeadTMaps 0  nextFreePage 0  nextFreeMap 5
tMapA[1]    tag 1  seqNo 1     primary mcell (vol,page,cell)  1 0 1   usr
tMapA[2]    tag 2  seqNo 1     primary mcell (vol,page,cell)  1 0 13   var
tMapA[3]    tag 3  seqNo 1     primary mcell (vol,page,cell)  2 2 4
ob_fset
#
```

Per volume bitfiles

Each volume is supported by these bitfiles:

- Reserved Bitfile Metadata Table (RBMT)
  - contains mcells for reserved bitfiles
  - last mcell in each RBMT page will link to the next page
  - eliminates BMT fragmentation problems

Displaying RBMT summary information (1 of 2)

```
# nvbmtpg -r -R usr_domain
================================================================
DOMAIN "usr_domain"  VDI 1 (/dev/rdisk/dsk2g)  lbn 32   RBMT
page 0
----------------------------------------------------------------
There is 1 page in the RBMT on this volume.
There are 19 free mcells in the RBMT on this volume.
================================================================
DOMAIN "usr_domain"  VDI 2 (/dev/rdisk/dsk4b)  lbn 32   RBMT
page 0
----------------------------------------------------------------
There is 1 page in the RBMT on this volume.
There are 19 free mcells in the RBMT on this volume.
```

Displaying RBMT summary information (2 of 2)

- Bitfile metadata table (BMT)
  - contains all the mcells for nonreserved bitfiles
  - grows as new files are created
- Storage bitmap (SBM)
  - contains 1 bit for every 8K bytes

    (1 bit per 1K in Tru64 UNIX V4.0)
- Misc bitfile
  - contains bootblocks, and so forth
  - four pages in size

Displaying BMT for volume 1 of usr_domain

```
# nvbmtpg -r usr_domain 1
================================================================
=
DOMAIN "usr_domain"  VDI 1 (/dev/rdisk/dsk2g)  lbn 48   BMT page
0
----------------------------------------------------------------
-
There are 1025 pages in the BMT on this volume.
BMT uses 2 extents (out of 33) in 2 mcells.
```

Per fileset bitfiles (1 of 2)

- Tag file (not `.tags`)
  - translates tag # into location of primary mcell (within BMT) for the appropriate file
  - formerly called the ‘Tag Directory File’
- Tags can be reused as files are deleted
  - limited to size of associated sequence number
  - 8001 is a typical sequence number showing that this tag is in use (left bit is set) and it is in use for the first time (001)
  - limits tag reuse to ~4k times before tag is dead

Per fileset bitfiles (2 of 2)

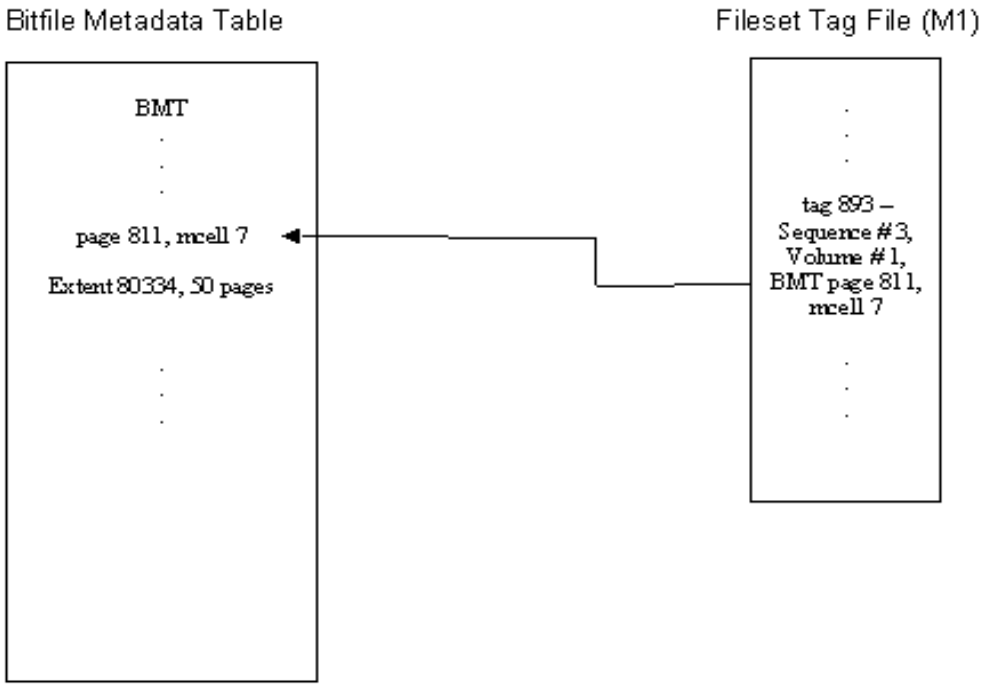
Tag file consists of 8K pages with 1022 tagmap entries

- (8 bytes each) preceded by a 16-byte header

Tagmap entry contains:

- Sequence number
- Volume index
- Primary Mcell ID (BMT page # and cell # within page)

Fileset tag directory locating primary mcell



Displaying a fileset tag file

```
# nvtagpg -r usr_domain usr -a
================================================================
DOMAIN "usr_domain"  VDI 1 (/dev/rdisk/dsk1g)  lbn 112   "usr" TAG page 0
----------------------------------------------------------------
numAllocTMaps 1021  numDeadTMaps 0  nextFreePage 30  nextFreeMap 0

tMapA[0] tmFreeListHead 50  tmUninitPg 50
tMapA[1]    tag 1        seqNo 1      primary mcell (vol,page,cell)  1 0 3
tMapA[2]    tag 2        seqNo 1      primary mcell (vol,page,cell)  1 0 4
tMapA[3]    tag 3        seqNo 1      primary mcell (vol,page,cell)  1 0 6
   (...)
tMapA[1021] tag 1021     seqNo 1      primary mcell (vol,page,cell)  1 40 23
================================================================
DOMAIN "usr_domain"  VDI 1 (/dev/rdisk/dsk1g)  lbn 128   "usr" TAG page 1
----------------------------------------------------------------
numAllocTMaps 1022  numDeadTMaps 0  nextFreePage 30  nextFreeMap 0

tMapA[0]    tag 1022     seqNo 1      primary mcell (vol,page,cell)  1 40 25
tMapA[1]    tag 1023     seqNo 1      primary mcell (vol,page,cell)  1 40 27
```

Finding primary mcell thru tag directory using tag number

```
# ls -li bigfile
 2324 -rwxr-xr-x   1 root     system   15429536 Nov 24 14:54
bigfile
# df .
Filesystem      512-blocks        Used   Available Capacity
Mounted on
var_domain#var      524288      273858      238416    54%    /var
# ntagpg -r var_domain var -t 2324
================================================================
DOMAIN "var_domain"  VDI 1 (/dev/rdisk/dsk1h)  lbn 144   "var" TAG page 2
----------------------------------------------------------------
numAllocTMaps 350  numDeadTMaps 0  nextFreePage 0  nextFreeMap 337

tMapA[280]  tag 2324     seqNo 49     primary mcell (vol,page,cell)  1 104 20
```

Finding primary mcell thru tag directory using file name

```
# ls -li bigfile
 2324 -rwxr-xr-x   1 root     system   15429536 Nov 24 14:54
bigfile
# pwd
/var/cluster/members/member0/tmp

# df .
Filesystem      512-blocks        Used   Available Capacity  Mounted on
var_domain#var      524288      273858      238416    54%    /var
# ntagpg -r var_domain var nvtagpg -r var_domain var \
            cluster/members/member0/tmp/bigfile
================================================================
DOMAIN "var_domain"  VDI 1 (/dev/rdisk/dsk1h)  lbn 144   "var" TAG page 2
----------------------------------------------------------------
numAllocTMaps 350  numDeadTMaps 0  nextFreePage 0  nextFreeMap 337

tMapA[280]  tag 2324     seqNo 49     primary mcell (vol,page,cell)  1 104
```

Fragment bitfile

- Contains small files including the last parts of small files
- Size varies with number of small files
- Has tag number 1

Reserved bitfile special names (1 of 3)

All accessible under .tags directory

Name	Function	One Per
M-6, M-12, ...	Reserved Bitfile Metadata Table	Volume
M-7, M-13, ...	Storage Bitmap	Volume
M-8, M-14, ...	Root Tag File	Domain
M-9, M-15, ...	Transaction Log	Domain
M-10, M-16, ...	Bitfile Metadata Table	Volume
M-11, M-17, ...	Miscellaneous Bitfile	Volume

Reserved bitfile special names (2 of 3)

Name	Function	One Per
M1	Fileset tag file (not .tags) for fileset #1.	Fileset
M2	Fileset tag file (not .tags) for fileset #2.	Fileset
M3	Fileset tag file (not .tags) for fileset #3.	Fileset
(...)		
Mn	Fileset tag file (not .tags) for fileset #n.	Fileset
1	Fragment Bitfile	Fileset

Reserved bitfile special names (3 of 3)

Name	Function	One Per
2	Fileset's Root Directory	Fileset
3	.tags Directory	Fileset
4	User Quota File	Fileset
5	Group Quota File	Fileset
6	User File with Tag # 6	Fileset
7	User File with Tag # 7	Fileset
(...)		
n	User File with tag # n	Fileset

Metadata bitfile tags

Tags for reserved bitfiles of virtual disk *i* are:

- **tag = - (*magic-number-shown-below* + (*vol_index* * 6))**

Metadata Bitfile Tags Example

Reserved File	Formula	Disk	Disk 2
RBMT	- (0 + (vol * 6))	-6	-12
SBM	- (1 + (vol * 6))	-7	-13
Root tag file	- (2 + (vol * 6))	-8	-14
Log	- (3 + (vol * 6))	-9	-15
BMT	- (4 + (vol * 6))	-10	-16
Misc Bitfile	- (5 + (vol * 6))	-11	-17

- **May be printed in weird tags which effectively translate to negative numbers**

.tags for directory entries for metadata bitfiles

Start with an `M`

- **For virtual disk specific files, use the negative number**
- **For fileset tagfiles, use the fileset ID**

For example, if `/usr` is an AdvFS fileset:

File	Description
`/usr/.tags/1`	`Fragment bitfile`
`/usr/.tags/M-6`	`RBMT of disk 1`
`/usr/.tags/-6`	`RBMT of disk 1 also`
`/usr/.tags/M-15`	`Log of disk 2`
`/usr/.tags/M2`	`tag file for 2nd fileset`

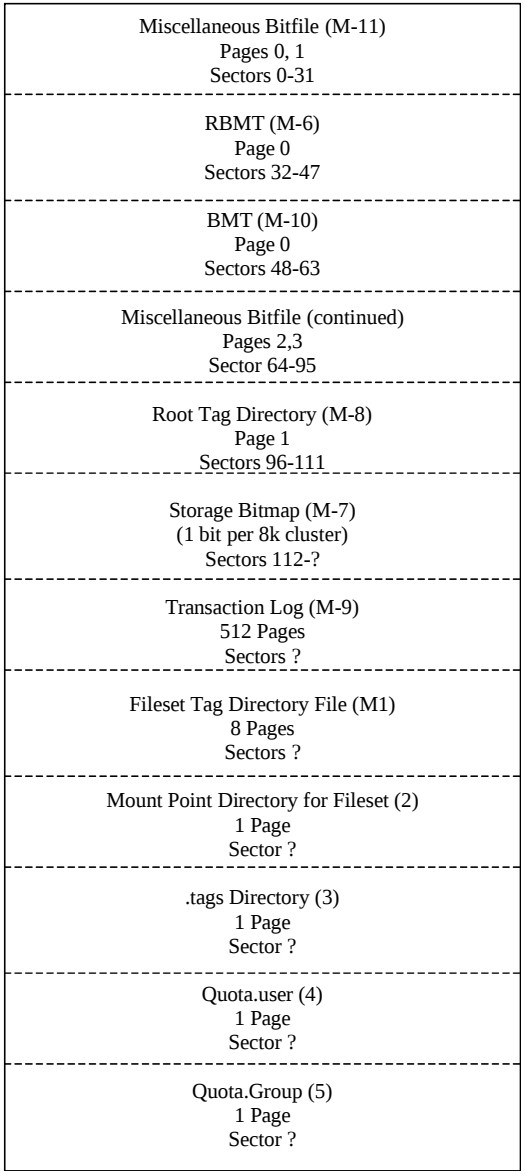
Bitfile metadata table (1 of 2)

- **BMT is represented by a file found under.`tags` directory**
  - special file name is M-10 (more later)
  - contains a series of 8K pages just like any other AdvFS file
  - pages contain mcells (292 bytes each) and header information
- **Each mcell contains one or more variable-length records**
  - describe various file attributes, extents, permissions, frag info, etc
- **BMT can grow just as any file can grow - just adds another extent**
  - starts with slightly more than 1M (can be tailored)
- **BMT is created when the `mkfdmn` command is issued**
  - one BMT for each volume in the domain

Bitfile metadata table (2 of 2)

- **Contains all mcells for all files other than the reserved bitfiles**
  - user files
  - user directories
- **Mcells for the reserved bitfiles are in RBMT**
- **RBMT and BMT:**
  - first mcell describes itself
  - as more mcells are needed, it grows using extents
  - RBMT reserves the last mcell on each page to chain to other pages of mcells

Reserved bitfiles on disk layout



Mcell records (1 of 2)

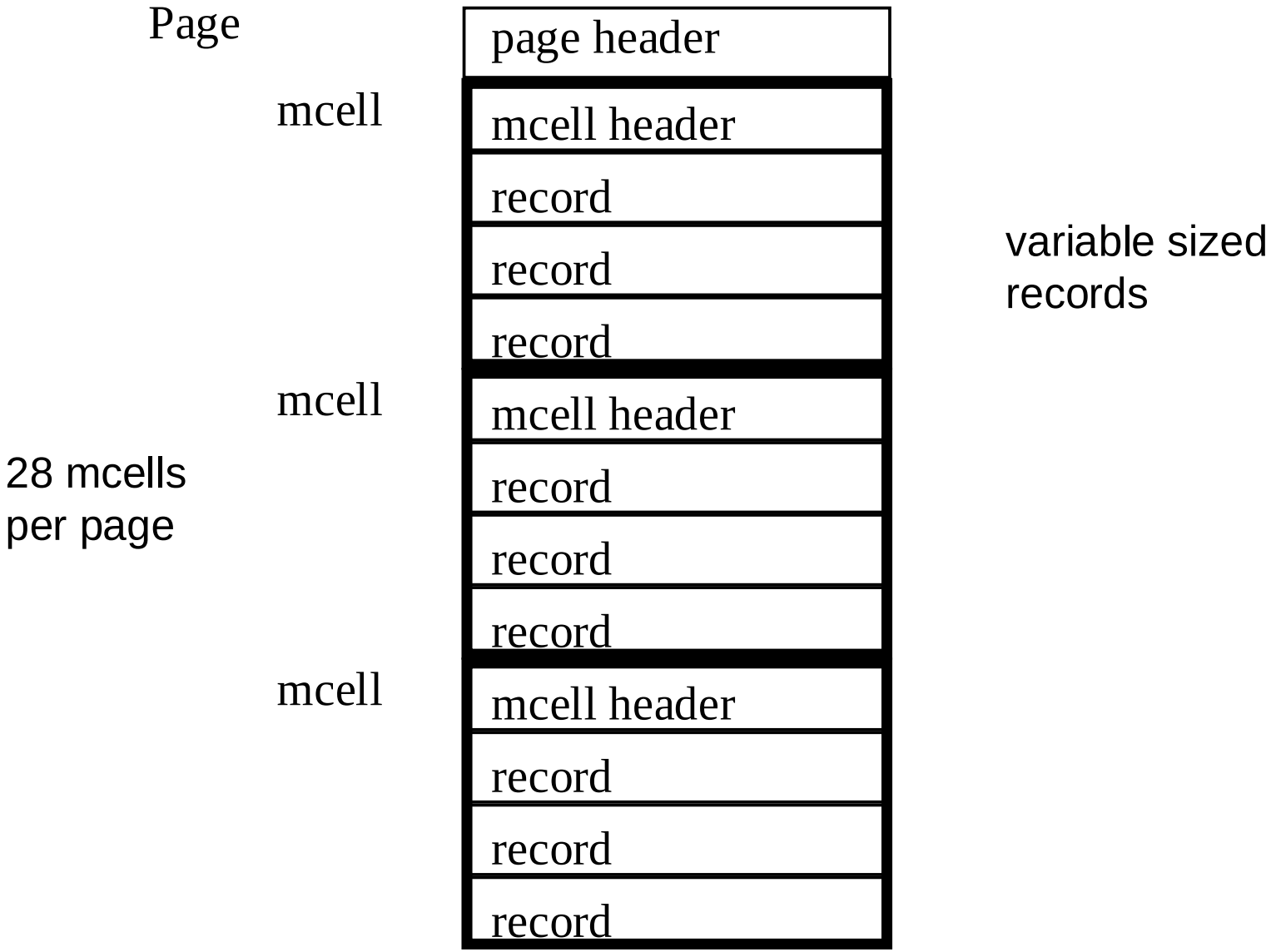
- **The inodes of AdvFS**
- **28 Fixed-size (292 byte) mcells are packed into 8K pages**
- **One or more linked mcells describe bitfiles**
  - First mcell in list is primary mcell
- **Each mcell contains variably sized records describing attributes of the bitfile**
- **Contained in the Bitfile Metadata Table (BMT) and the RBMT**

Mcell records (2 of 2)

Record Types include:

- Extent Maps (of various kinds)
- Bitfile Attributes (clone, original, etc.)
- Domain Attributes
- Virtual Disk Attributes (disk ID, disk index, etc.)
- Fragment Attributes - more later
- POSIX file stats (permissions, size, link count, etc.)
- Symbolic link targets

Mcell page structure



RBMT page 0

RBMT Page 0:

- **Starts at sector 32 (LBN 32)**
- **Contains these primary mcells:**
  - mcell 0 Reserved Bitfile Metadata Table (RBMT)
  - mcell 1 Storage Bitmap (SBM)
  - mcell 2 Root Tag File - Optional, one per domain
  - mcell 3 Log - Optional, one per domain
  - mcell 4 Bitfile Metadata Table (BMT)
  - mcell 5 Misc bitfile

BMT page 0

- **Starts at Sector 48**
- **mcell 0 is head of the BMT page free list**
- **Contains mcells for nonreserved bitfiles, such as user files and directories**
- **All other BMT pages are found via the RBMT**

BMT page format

Each 8192-byte page is composed of:

- 16-byte header followed by
- 28 292-byte mcells

BMT header consists of:

- Pointer to next free mcell on page
- Pointer to next page with free mcells
- Number of free mcells on the page
- Page number (within BMT)
- AdvFS version (now 4)

Mcell addresses

- **Mcells are addressed by a 32-bit mcell ID, `bfMCIdT`**
  - 27 bits give the mcell's BMT page number
  - 5 bits give the mcell's position within its page
- **Every bitfile has a primary mcell**
- **Tagfiles map (tag file)**
  - tag numbers can be mapped to primary mcell locations

Reserved mcell addresses

Within RBMT page 0, the slot numbers listed are used to calculate the tag numbers for the reserve bitfiles:

- **0: the RBMT itself**
- **1: SBM, storage bitmap**
- **2: root tag file**
- **3: transaction log**
- **4: BMT**
- **5: misc bitfile**

Mcell format

- **Begins with 24-byte header**
  - 32-bit ID of the next mcell in the chain and virtual disk containing next mcell
  - position of this mcell within the chain
  - tag number of the bitfile
  - tag number of the fileset
- **Has 268 bytes remaining for mcell records**

Mcell records

- **Vary in size and type**
- **Begin with a 4-byte header**
  - 2-byte count of record size
  - 1-byte type field (There are currently about 20 record types)
  - 1-byte version number for different versions of a type
- **Have a null record whose size is 4 and type is 0**
  - this is the end-of-records indicator

Utilities for viewing records

- **nvbmtpg**
- **nvfragpg**
- **nvlogpg**
- **nvtagpg**
- **vsbmpg**
- **vfilepg**
- **savemeta**

Extent maps for nonreserved files

For nonreserved files:

- **Primary extent map record**
  - within the primary mcell
  - allocated when file gets a page
  - if full, points to an extra extent map
- **Extra extent map record**
  - allocated as the file grows

For striped files, extent map records use a shadow extent map and may point to more than one disk.

Extent maps for reserved files

For reserved bitfiles, for example, the RBMT

- Primary extent map
  - if full, points to extra extent map
- Extra extent map
  - usually only needed for the BMT itself

All records for reserved files are in the RBMT

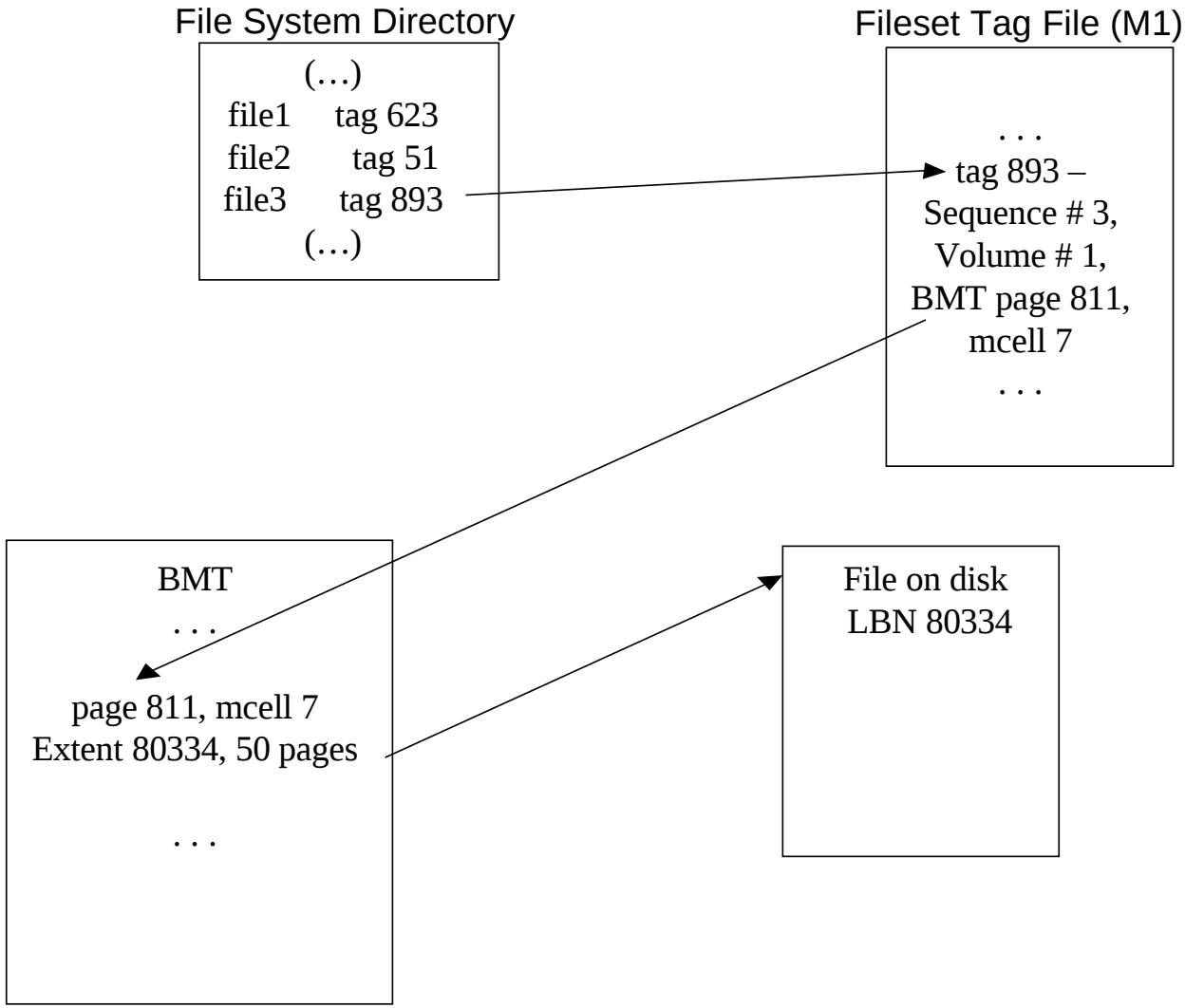
Encoding of extents

- **Extents information is captured in two fields**
  - page number within bitfile
  - block number within virtual disks
- **Size of one extent is inferred from the next**
  - compute difference between the page numbers
  - place -1 in block number to indicate a hole (-2 for clone hole)
- **Try the following:**
  - page = 0, block = 8000
  - page = 100, block = -1
  - page = 200, block = 9600
  - page = 300, block = -1

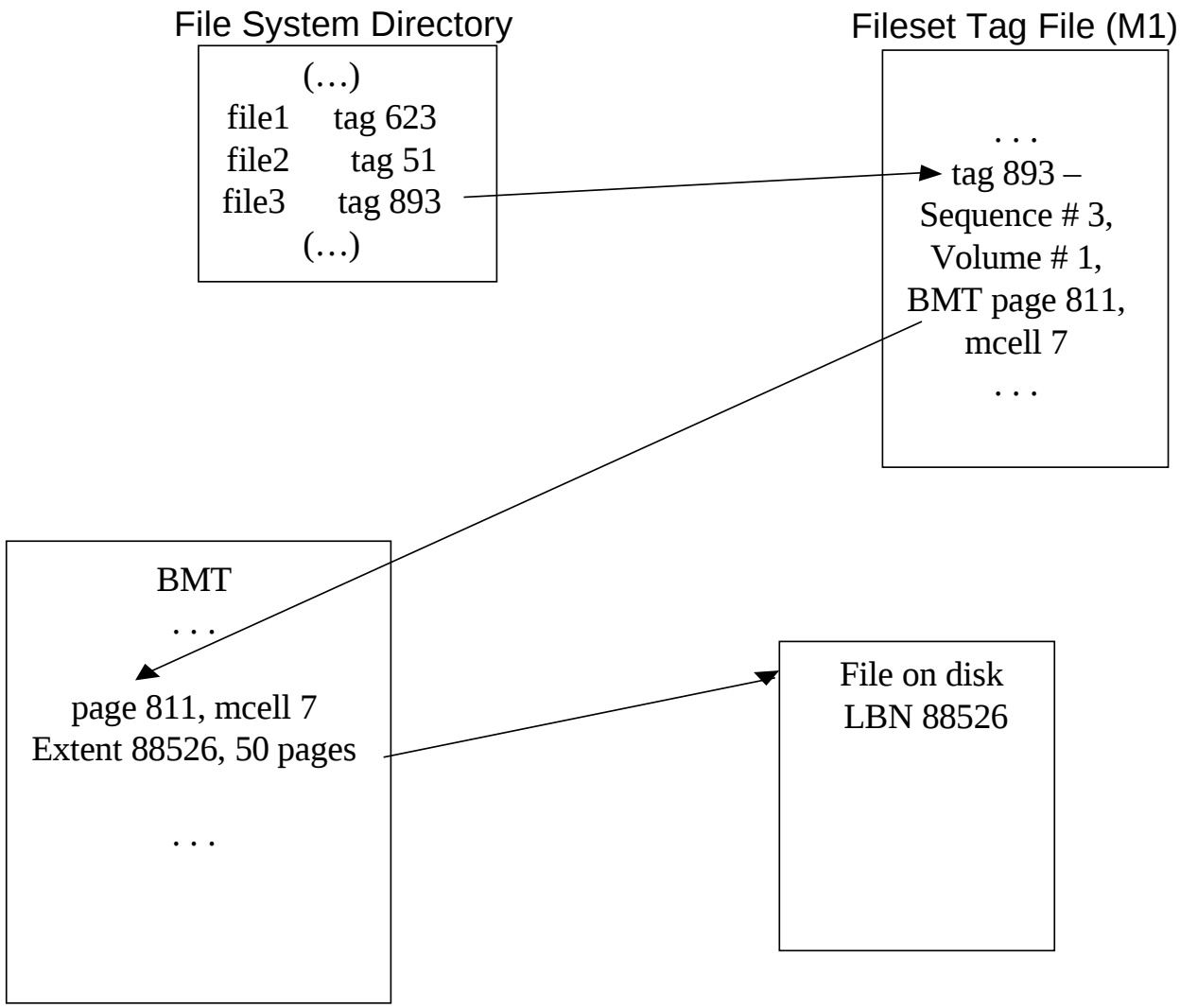
Tag file characteristics

- **Bitfiles are identified by tags**
- **Tag files are:**
  - arrays of tagmap entries
  - indexed by tag number
- **Tag File entries contain:**
  - sequence number (high bit set if in use)
  - volume index
  - mcell ID within volume
- **Justifying Tag Files**
  - tag files introduce an extra level of overhead in accessing file data

File access through tag file



Tag file allowing transparent data move



Tag file page

- **Starts with a five-field, 16-byte header**
  - number of this page (for sanity checking)
  - next page with free tagmap entries
  - next free tagmap within this page
  - number of of allocated tagmaps
  - number of dead tagmap entries
- **Followed by 1022 tagmap entries**

Tagmap entries

- **Head of free tagmap list**
  - stored in first slot of page 0
  - points to:
    - first page with a free tagmap entries
    - first uninitialized page
- **Element of free tagmap list**
  - sequence number
  - pointer to next free entry on this page
- **Allocated tagmap entry**
  - sequence number
  - virtual disk of bitfile
  - mcell ID of bitfile

Root tag file

- **Entries for each fileset of the domain**
- **Contains mcell IDs of fileset tagfiles**
- **Can be used to find the list of domain filesets**

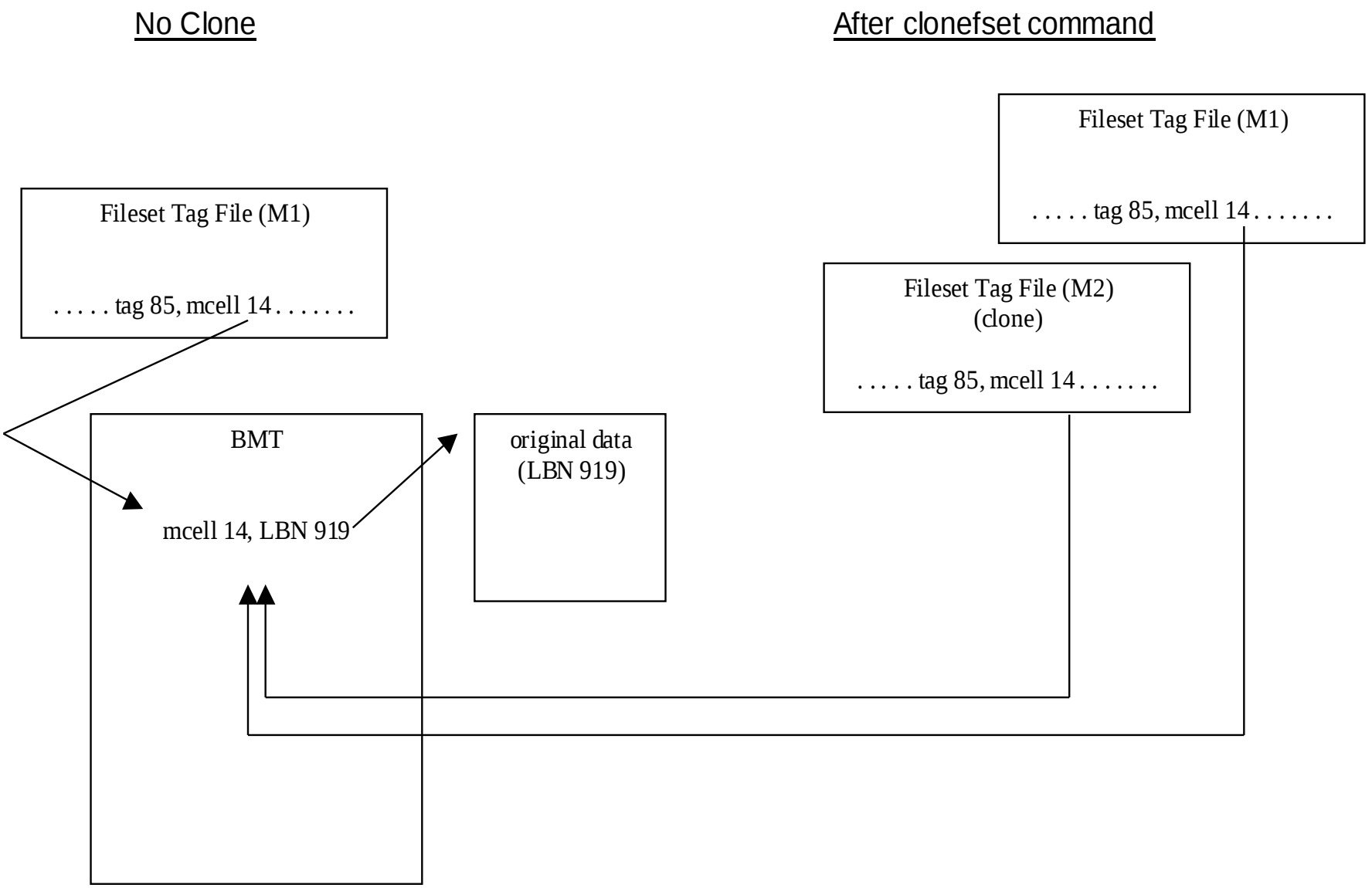
Fileset tag file

- **Each fileset has its own tag file**
- **Maps fileset's bitfiles to primary mcells**
- **Has these special fileset tags**
  - 1 Fragment bitfile
  - 2 Root directory
  - 3 .tags
  - 4 User quota file
  - 5 Group quota file

Cloning through fileset tag

- **Clone fileset is a read-only virtual copy of the data as it existed at the time the clone was created**
  - should be used as source for a backup operation and then removed
  - creates a copy of the Fileset Tag File (M1)
  - still points to the same BMT
- **As original data changes while clone exists:**
  - copy of the original data must be created
  - new mcell must be allocated in the BMT
  - appropriate entry in new copy of Fileset Tag file must be updated to point to cloned page's new mcell in BMT

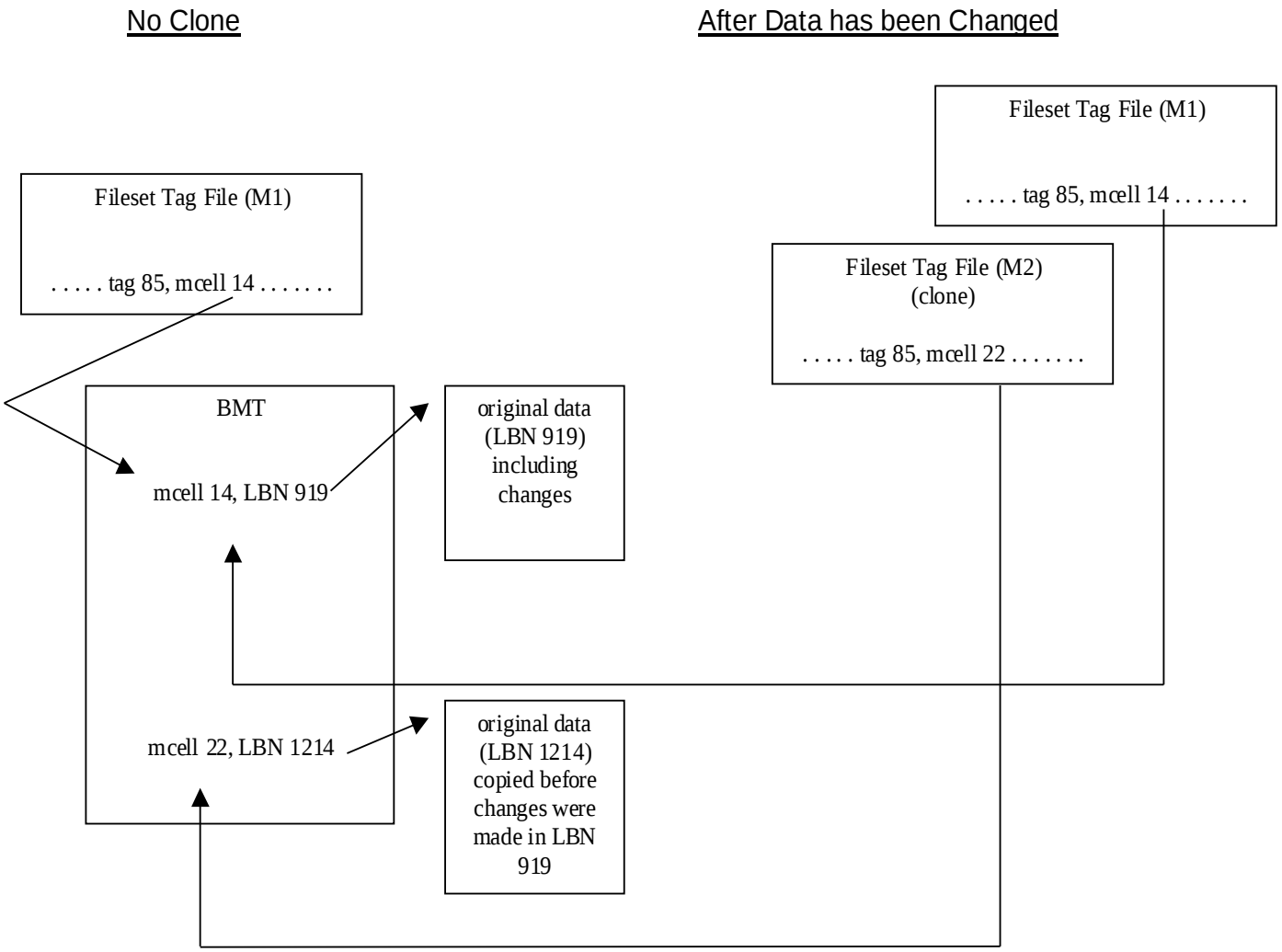
Fileset tag file before and after cloning



Fileset tag file before and after cloning

- **No data is copied unless a change is made to original data**
- **A copy is only made of data that is about to be changed**
- **Clone is effectively a 'snapshot' of the data at a known time**

Clone structures after data write



Utility for viewing tag files

The `nvtagpg` utility:

- **Prints formatted pages of a root tag file or a fileset tag file**

Displaying root tag file information

```
# nvtagpg -r usr_domain


================================================================
DOMAIN "usr_domain"  VDI 1 (/dev/rdisk/dsk2g) lbn 96  root TAG page 0
----------------------------------------------------------------
currPage 0
numAllocTMaps 3  numDeadTMaps 0  nextFreePage 0  nextFreeMap 5
tMapA[1]    tag 1  seqNo 1     primary mcell (vol,page,cell)  1 0 1
usr
tMapA[2]    tag 2  seqNo 1     primary mcell (vol,page,cell)  1 0 13
 var
tMapA[3]    tag 3  seqNo 1     primary mcell (vol,page,cell)  2 2 4
ob_fset
```

Displaying fileset tag file

Given a particular tag number, divide by 1022:

- **Quotient is the page**
- **Remainder is tagmap slot**

```
# nvtagpg -r usr_domain usr


================================================================
DOMAIN "usr_domain"  VDI 1 (/dev/rdisk/dsk2g) lbn 1047552  "usr" FRAG page 0
----------------------------------------------------------------
currPage 0
numAllocTMaps 1021  numDeadTMaps 0  nextFreePage 23  nextFreeMap 0
tMapA[1]    tag 1    seqNo 1     primary mcell (vol,page,cell)  1 0 3
tMapA[2]    tag 2    seqNo 1     primary mcell (vol,page,cell)  1 0 4
tMapA[3]    tag 3    seqNo 1     primary mcell (vol,page,cell)  1 0 6
tMapA[4]    tag 4    seqNo 1     primary mcell (vol,page,cell)  1 0 7
tMapA[5]    tag 5    seqNo 1     primary mcell (vol,page,cell)  1 0 8
(...)
```

Displaying individual file's tag file entry

```
# ls -li big1
22896 -rwxr-xr-x   1 root     system   13729520 Jun 24 16:53 big1
#
# nvtagpg -r usr_domain -T 1 -t 22896
==============================================

DOMAIN "usr_domain"  VDI 1 (/dev/rdisk/dsk2g)  lbn 2055136
"usr" FRAG page 22

----------------------------------------------

currPage 22
numAllocTMaps 1022  numDeadTMaps 0  nextFreePage 0  nextFreeMap 0
tMapA[412]  tag 22896    seqNo 1     primary mcell (vol,page,cell)  1
951 15
# bc
22896/1022
22
22896%1022
412
^D
```

UNIX directories

- **UNIX directories are contained in standard bitfiles**
  - Directories are a series of directory entries
- **POSIX directory entry format**
  - inode number: 32 bits
  - entry length: 16 bits
  - name length: 16 bits
  - name: variable-length, zero-padded to fill the entry
- **AdvFS directory information *hidden* in POSIX entry**
  - tag numbers replace inode numbers
  - the 64-bit `tag.sequence` ID is placed in the name “padding”

AdvFS directory index files

File name lookup through large directory files is very slow

- On average, one-half of the directory entries must be searched

AdvFS uses a B-tree to index files to speed the search

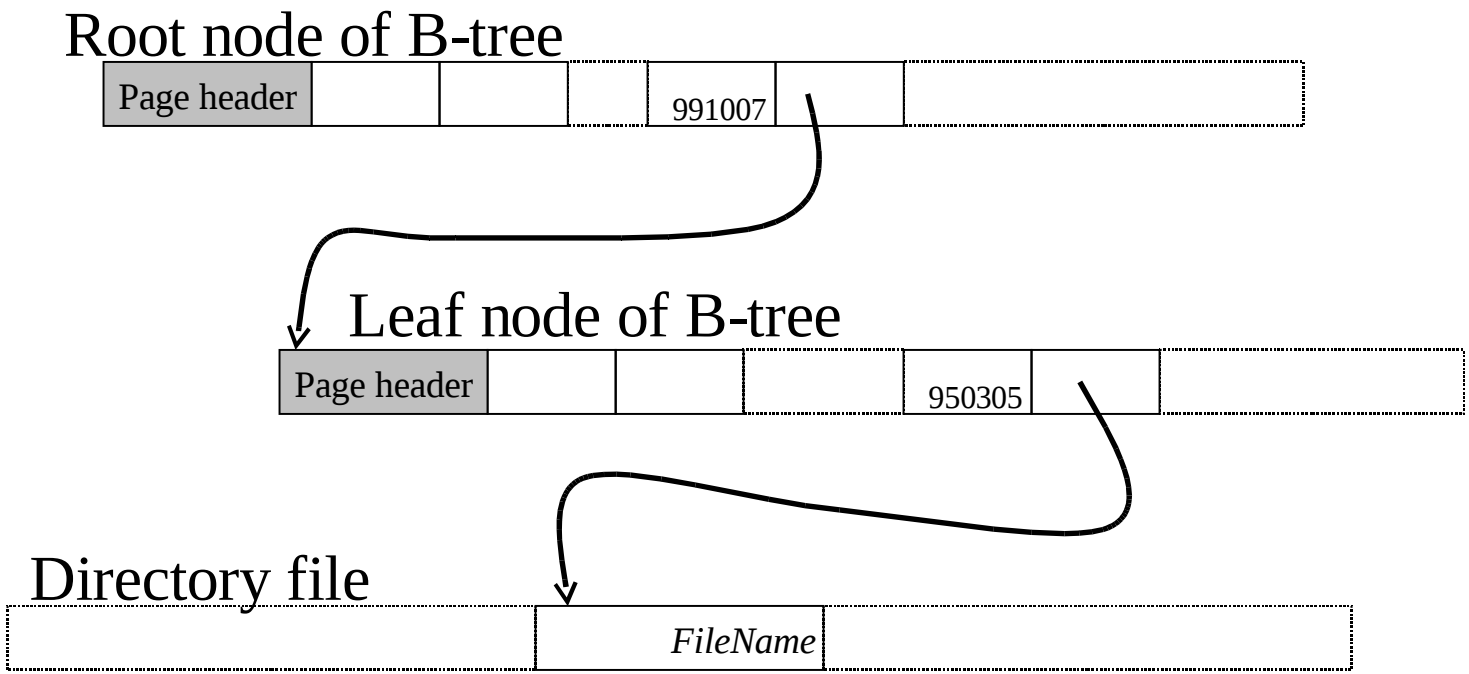
- Each page of the index file has 510 entries
- Filenames are “hashed” using ELF
  - b-tree maps hash keys to directory entry locations
- Mcell chain of large directory
  - contains tag.sequence number of the index file
- Mcell chain of the index file
  - identifies the height and root of the B-tree

Displaying the location of the index file

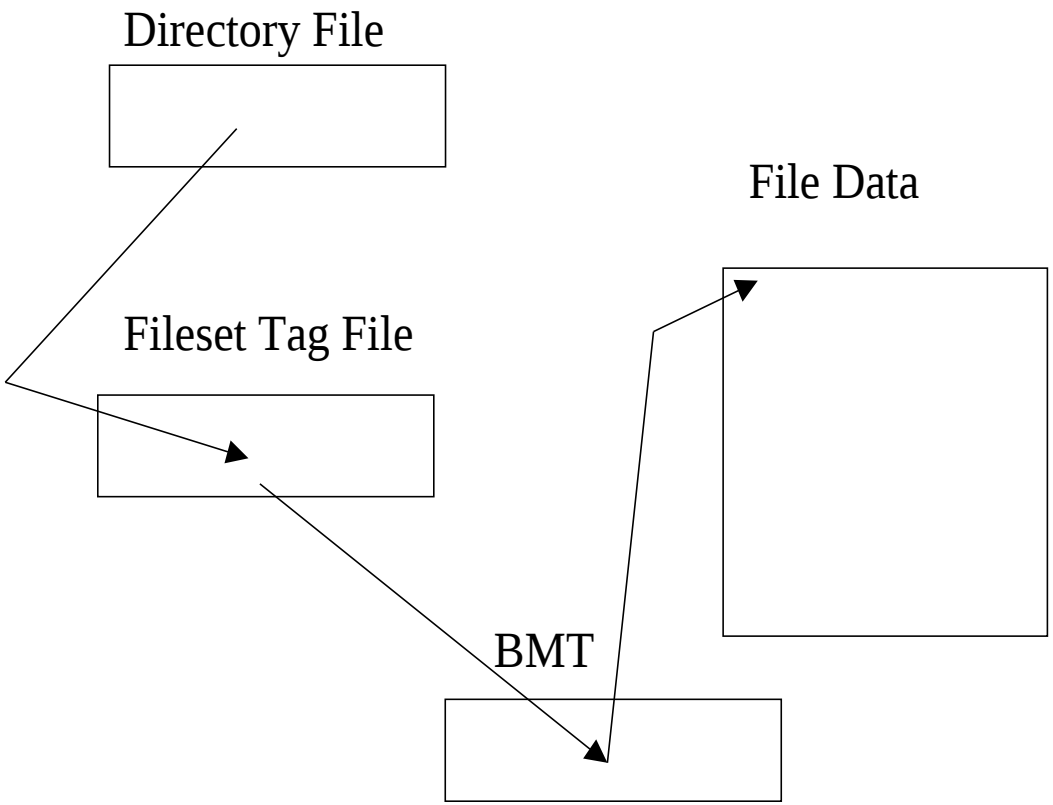
```
# showfile -i LotsOfFiles


          Id  Vol  PgSz  Pages  XtntType  Segs  SegSz  I/O
Perf  File
   713.8001    1    16      4    simple    **     **  ftx
33%  index ()
# ls -ld LotsOfFiles
drwxr-xr-x   2 root     system     32768 Nov 22 18:20
LotsOfFiles
```

AdvFS directory index b-tree

ELF hash function maps file name *FileName* to 950305



Relationship to POSIX files



AdvFS tagfiles and migration

The ability to move:

- A bitfile's data transparently - this is supported by the buffer cache and I/O scheduling algorithms
- A bitfile's metadata to another volume - tag files enable this

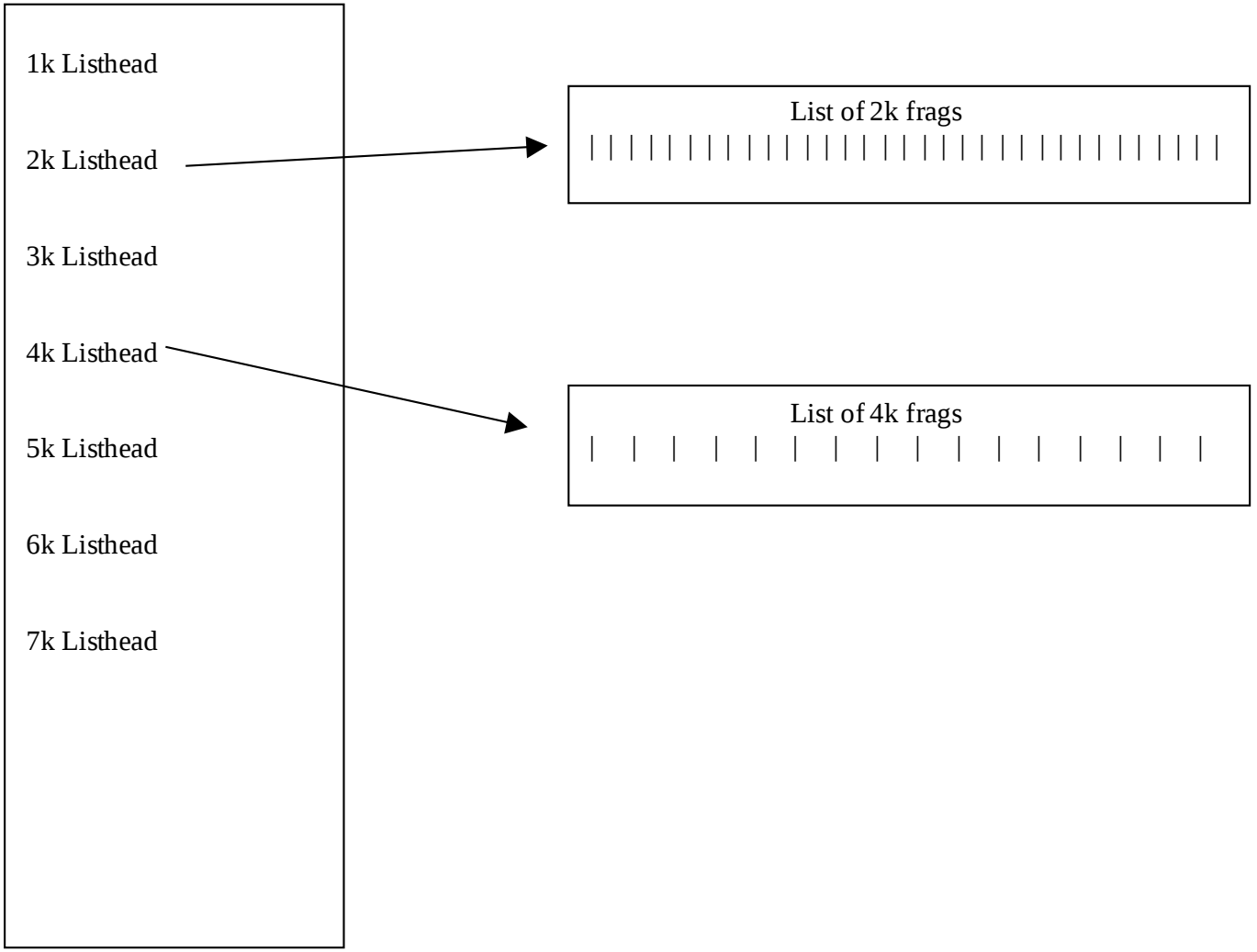
Fragment bitfile

- One per fileset
- Contains small (< 8K) ends of files
- Bitfiles are allocated in 8K (16 sector) units
- If file is less than 100K, a fragment is used if needed.
- Fragment bitfile is broken into a series of 128KB fragment groups

Fragment groups

- **Each group consists of fragments of a particular size**
  - free (0K), 1K, 2K,...7K
- **Each fragment group has the form**
  - one page fragment header
  - fragments sized for the group
- **Fragment addresses are usually**
  - in 1K pages
  - relative to start of fragment bitfile
- **Fragments are not Extents!!**
  - extent is a contiguous range of 8K pages
  - fragment is a chunk of disk space between 1K and 7K in size

Fragment bitfile
locating fragment groups



Fragment header

- Pointer to next fragment group of this type with free space
- Page number of this page (a sanity check)
- Type of this group
- Number of free fragments
- Fileset ID
- Version
- List of free fragments in the group

Fragment group statistics display

```
# nvfragpg -r usr_domain usr
================================================
DOMAIN "usr_domain"
------------------------------------------------
reading 438 frag group headers, 400 headers read
```

frag type	free	1K	2K	3K	4K	5K	6K	7K	totals
groups	2	44	67	80	76	63	52	54	438
frags	-	5588	4254	3386	2413	1600	1100	979	19320
frags used	-	5469	4245	3364	2405	1581	1093	973	19130
disk space	256K	5632K	8576K	10.0M	9728K	8064K	6656K	6912K	54.8M
space used	-	5469K	8490K	9.9M	9620K	7905K	6558K	6811K	53.7M
space free	254K	119K	18K	66K	32K	95K	42K	42K	668K
overhead	2K	44K	67K	80K	76K	63K	52K	54K	438K
wasted	-	0K	67K	80K	228K	126K	52K	54K	607K
% used	-	97%	98%	98%	98%	98%	98%	84%	98%

Fragment free list display using nvfragpg (1 of 3)

```
# nvfragpg -fr usr_domain usr
=======================================================
DOMAIN "usr_domain"
-------------------------------------------------------
reading 438 frag group headers, 100 headers read
        reading 438 frag group headers, 200 headers read
        reading 438 frag group headers, 300 headers read
        reading 438 frag group headers, 400 headers read
```

frag type	free	1K	2K	3K	4K	5K	6K	7K	totals
groups	2	44	67	80	76	63	52	54	438
frags	-	5588	4254	3386	2413	1600	1100	979	19320
frags used	-	5469	4245	3364	2405	1581	1093	973	19130
disk space	256K	5632K	8576K	10.0M	9728K	8064K	6656K	6912K	54.8M
space used	-	5469K	8490K	9.9M	9620K	7905K	6558K	6811K	53.7M
space free	254K	119K	18K	66K	32K	95K	42K	42K	668K
overhead	2K	44K	67K	80K	76K	63K	52K	54K	438K
wasted	-	0K	67K	80K	228K	126K	52K	54K	607K
% used	-	97%	98%	98%	98%	98%	98%	84%	98%

Fragment free list display using nvfragpg (2 of 3)

```
head of free lists of frag groups from file set attributes:
frag type BF_FRAG_ANY  firstFreeGrp   6976  lastFreeGrp     32
frag type BF_FRAG_1K   firstFreeGrp   6960  lastFreeGrp    560
frag type BF_FRAG_2K   firstFreeGrp   6880  lastFreeGrp   6880
frag type BF_FRAG_3K   firstFreeGrp   6944  lastFreeGrp    848
frag type BF_FRAG_4K   firstFreeGrp   6832  lastFreeGrp    816
frag type BF_FRAG_5K   firstFreeGrp   6896  lastFreeGrp    864
frag type BF_FRAG_6K   firstFreeGrp   6768  lastFreeGrp   6768
frag type BF_FRAG_7K   firstFreeGrp   6864  lastFreeGrp   6864

any        6976   6992
free 1K    6960   6912   6928    560
full 1K       0    112    128    176    224    320        400    496    944
 1184
           1616   1840   2928   3184   3280   3424        3568   3808   3856
  3904
           3936   3968   4192   4544   4816   4864        4912   4928   4944
  5120
           5344   5456   5536   5600   5760   6240        6528   6608   6640
  6848
```

Fragment free list display using nvfragpg (3 of 3)

```
free 2K    6880
full 2K      64     208     256     272     288     336   480     624     784     928
            1040    1168    1264    1520    1664    2480  2736    3024    3056    3072
            3088    3104    3120    3200    3360    3504  3792    3824    3872    3888
            3952    4032    4080    4144    4160    4288  4304    4352    4400    4448
            4496    4528    4560    4592    4608    4624  4656    4688    4704    4896
            4976    5168    5360    5488    5552    5584  5616    5664    5824    5984
           6144    6320    6480    6560    6624          6656
(...)
free 7K    6864
full 7K      80     384     512     544     656     704   768     832     912    1024
            1328    1440    1568    1680    1792    2528  2608    2688    2800    2944
            3168    3296    3408    3488    3536    3584  3600    3632    3664    3680
            3712    4048    4224    4432    4784    5072  5184    5200    5232    5264
            5472    5680    5792    5888    6000    6080  6128    6224    6336    6400
            6448    6672    6816
```

Fragments and files

- **POSIX stats record of mcells contains**
  - `fragId` field
    - `fragId.frag` is the page offset of fragment
    - `fragId.type` is the size of fragment
- **Use `showfile` to determine if there is a fragment**
  - does number of pages match file size?
- **Use `nvbmtpg` to find fragment location**

Tracking down a fragment (1 of 2)

```
# ls -i /etc/disktab
 1931 disktab
# nvbmtpg -r root_domain root -t 1931 -c
================================================================
DOMAIN "root_domain"  VDI 1 (/dev/rdisk/dsk1a)  lbn 222464   BMT page 82
----------------------------------------------------------------
pageId 82  megaVersion 4
freeMcellCnt 0    nextFreePg -1   nextfreeMCId page,cell 0,0
CELL 25       next mcell volume page cell 0 0 0        bfSetTag,tag 1,1931

          ..................................

RECORD 2   bCnt 92                                          BMTR_FS_STAT
st_mode 100755 (S_IFREG)   st_uid 3   st_gid 4   st_size 30910
st_nlink 1   dir_tag 13  st_mtime Wed May 29 07:06:42 2002
fragId.type BF_FRAG_7K  fragId.frag 2396
```

Tracking down a fragment (2 of 2)

```
# dd if=/.tags/1 of=/tmp/dtfrag bs=1024 \
                                iseek=2396 count=7
7+0 records in
7+0 records out
# head /tmp/dtfrag
#8192:fc#1024:\
        :od#393216:pd#324698:bd#8192:fd#1024:\
        :oe#717914:pe#324698:be#8192:fe#1024:\
        :of#1042612:pf#324698:bf#8192:ff#1024:\
        :og#393216:pg#819200:bg#8192:fg#1024:\
```

Storage bitmap bitfile characteristics

- **Storage bitmap file represents each 8K of on-disk storage within a volume with 1 bit (1K per bit in Tru64 UNIX V4.0)**
- **Basically a bunch of bits representing whether storage is in use or not**
- **SBM file is `.tags/M-7`.**
- **One per volume**
- **All storage is either free or in a bitfile**
- **AdvFS storage is allocated in pages**
- **On-disk SBM**
  - little more than an array of bits (1 bit per page)

SBM format

- **SBM page header**
  - sequence number, now unused (32 bits)
  - XOR field for rest of page (32 bits)
- **Bitmap**
  - 65472 bits
  - enough for 8184 pages (8k each)

SBM display through .tags/M-7 (1 of 2)

```
# od -x -N 1000000 /facusers/.tags/M-7
0000000  0000 0000 ffff 0000 ffff ffff ffff ffff
0000020  ffff ffff ffff ffff ffff ffff ffff ffff
*
0000120  ffff ffff ffff ffff ffff 0000 0000 0000
0000140  0000 0000 0000 0000 0000 0000 0000 0000
*
0020000  0000 0000 0321 4ed9 0000 0000 0000 0000
0020020  0000 0000 0000 0000 0000 0000 0000 0000
*
0033340  0000 f800 ffff f1ff ffff e7ff ff7f ffff
0033360  ffff 9fff 3fff fc00 7fff 8020 ffff f7ff
0033400  dfff ffff ffff ffff 1fcf 7ffc ffff f7ff
0033420  0fff fffe d0df fffd fff7 ffff ffff 7ffb
0033440  fff8 ffff ffff ffff 7fff 0000 0020 0000
0033460  8000 0002 2040 0000 0040 0000 0000 0002
0033500  0000 0000 0000 0000 0000 0000 0000 0000
```

SBM display through .tags/M-7 (2 of 2)

```
# showfile -x /facusers/.tags/M-7

        Id  Vol  PgSz  Pages  XtntType  Segs  SegSz  I/O   Perf  File
fffffff9.0000    1    16      9    simple    **     **  ftx   100%  M-7

    extentMap: 1
        pageOff    pageCnt     vol    volBlock    blockCnt
              0          4       1         112          64
        extentCnt: 1
#
# ls -l /facusers/.tags/M-7
----------   0 root     system     73728 Dec 31  1969 /facusers/.tags/M-7
```

SBM page allocation status display using vsbmpg

```
# vsbmpg -r facusers_dmn 1 -B 40000
================================================================================
DOMAIN "facusers_dmn"  VDI 1 (/dev/rdisk/dsk6c)  lbn 112   SBM page 0
--------------------------------------------------------------------------------
blocks 40000 - 40015 (0x9c40 - 0x9c4f) are mapped by SBM map entry 78, bit
4
   mapInt[78]  00000000 00000000 00000000 00000000
  block 40000
```

Miscellaneous bitfile

Characteristics of the misc bitfile includes:

- **One per volume**
- **Holds pages for**
  - primary and Secondary Boot Block
  - partition Table (Disk Label)
  - fake UFS Super Block with AdvFS Magic Number

Learning check

Lab 2


hp
invent