File System Corruptions & Best Practices

Karthik Iyer
Spectrum Scale Developement

Agenda

- GPFS On-disk Data Structures & Corruptions
- FSCK – Behind the Scenes
- Best Practices

Corruption Causes

- Loss of uncommitted data (including journal)
- Disk failure / overwrite
- Faulty hardware / firmware
- Cosmic rays; Solar flares
- Sometimes bugs in GPFS ;)

GPFS On-disk Data Structures
&
Corruptions

On-disk Data Structure Types

Cluster Configuration Data

Reserved files
- Inode file
- Block Allocation map file
- Inode Allocation map file
- Fileset Metadata file
- Log file
- ACL file
- Quota files
- Policy file
- Allocation Summary file

Descriptors
- NSD
- Disk
- File System

Other Metadata
- Directories
- Indirect Blocks
- Extended Attribute Overflow Blocks

Cluster Configuration Data

- Nodes, Configuration Parameters, NSDs, File Systems, Remote Clusters
- Latest version on quorum nodes (CCR) or primary/backup config servers
- Stored in flat file in node local path

  `/var/mmfs/gen/mmsdrfs`

- Can be lost due to OS reinstall, accidental deletion
- Corruption Symptoms
  - mmgetstate shows 'unknown' gpfs state
  - “This node does not belong to a GPFS cluster”
- Repair

  `mmsdrrestore -p remoteNode -F remoteFile [-a]`

  `mmccr backup -A archivePath` / `mmsdrbackup user exit`

Descriptors

- Fixed sectors on disk
- V1 – invisible to OS partitioning tools
- V2 – GUID Partition Table; 4K sector aligned

NSD V1 Sector	Descriptors	NSD V2 Sector
0	Partition Table	0-48
2	NSD	64-96
1	Disk	96-128
	Pdisk	128-160
8-4090	SG	2K-64K
4090-4098	Paxos	64K-72K
	Reserved	72K-128K

Descriptors (cont.)

- NSD (mmcrnsd)
  - identifies a block device as a GPFS NSD (mmdevdiscover)
  - NSD ID: 64 bit IP Addr + timestamp
- Disk (mmadddisk)
  - identifies file system to which NSD belongs
- File System (mmcrfs)
  - configuration like Name, BlockSize; Disks; Log Files; Global Snapshots; Storage Pools
  - read and write quorum of disks

```
disk         driver   sector     failure holds    holds                                storage
name         type       size       group metadata data  status        availability disk id pool         remarks
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------- ------------ ---------
nsd17        nsd         512          40 Yes      Yes   ready         up                 1 system        desc
nsd19        nsd         512          41 Yes      Yes   ready         up                 2 system        desc
nsd20        nsd         512          42 Yes      Yes   ready         up                 3 system        desc
nsd11        nsd         512          40 Yes      Yes   ready         up                 4 system
nsd14        nsd         512          41 Yes      Yes   ready         up                 5 system
nsd16        nsd         512          42 Yes      Yes   ready         up                 6 system
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
```

Descriptors (cont.)

- Can be lost due to accidental disk re-use/format
- Corruption Symptoms
  - mmlsdisk shows availability as "down"
  - mmfsadm test readdescraw <device>

    “No NSD descriptor in either sector 2 or 64 of <device>”
- Repair

```
# dd reserved sectors from a good disk of same file system
mmlsnsd -X # get NSD ID of bad disk
tspreparedisk -F -n <device> -p NSDid # patch in the NSD ID into NSD descriptor
mmfsadm test readdescraw <device> # verify descriptors
# NSD V2 supports backup (-B) and restore (-R)
tspreparedisk -[B|R] /tmp/desc.copy <device>
```

Reserved Files

- Contain metadata records
  - use reserved inode numbers (except root directory, log files)
  - not shown in directory tree
  - follow metadata replication settings
- Few Corruption Symptoms
  - FSSTRUCT errors in system log
  - file system panic; deamon Assert
  - log recovery failure; mount failure
  - admin commands fail (mmdeldisk)
- Repair
  - offline FSCK
  - sometimes manual recovery using tsdbfs
  - online replica compare if only some replicas are corrupt

MMFS_FSSTRUCT

- /var/log/messages

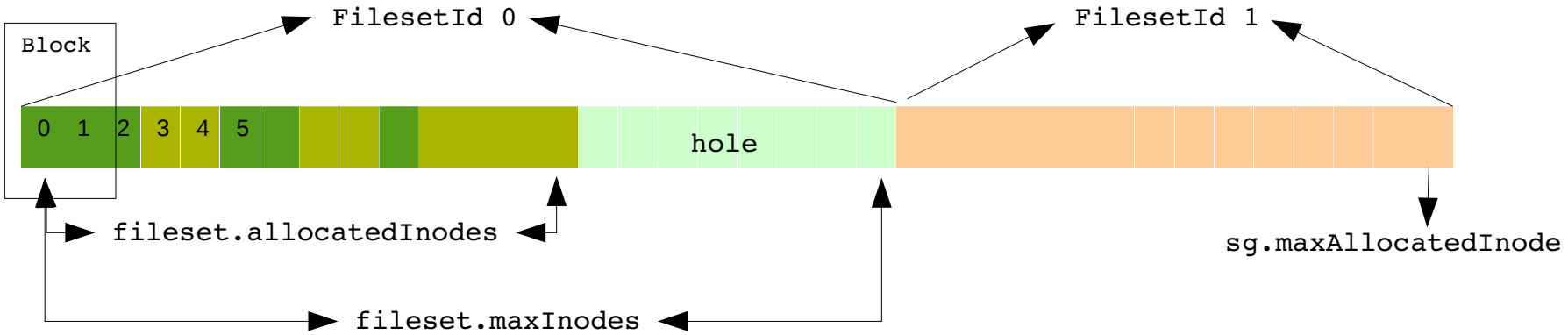
```
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:   Invalid disk data structure.
Error code 1108.   Volume fs1                                                          . Sense Data
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:    04 54 00 01 00 00 00 03
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:    00 00 00 00 11 7F E0 00
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:    00 08 00 01 00 00 00 03
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:    00 00 00 00 11 7F E0 00
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:    00 00 10 00 00 00 00 00
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:    00 00 00 04 09 00 20 03
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:    00 00 00 00 00 00 00 00
...
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:    00 00 00 00
Jan 1 05:54:36 node1 mmfs: Error=MMFS_FSSTRUCT, ID=0x94B1F045, Tag=13386052:
```

- /usr/lpp/mmfs/samples/debugtools/fsstructlx.awk

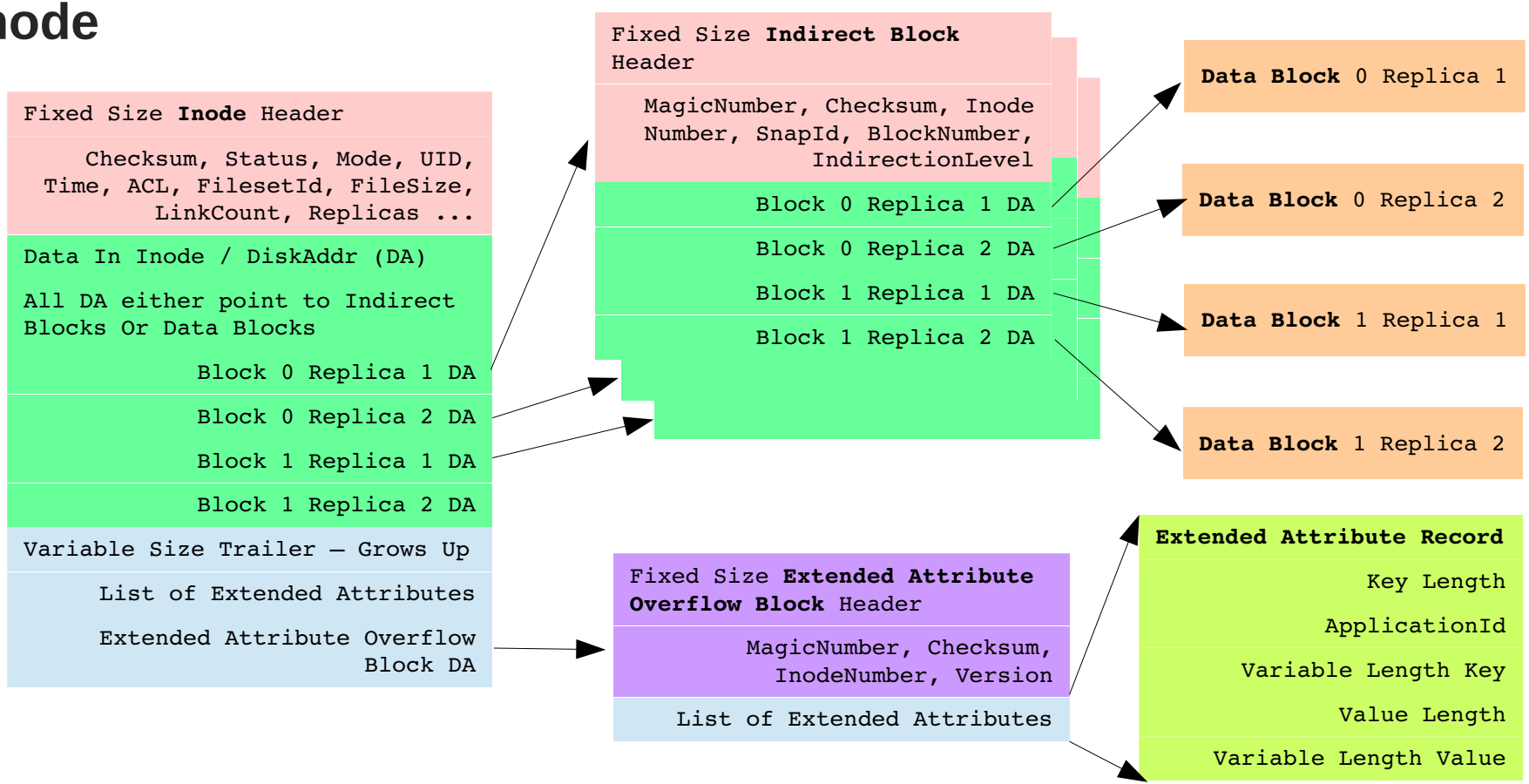
```
01/01@05:54:36 node1 FSSTRUCT fs1 108 FSErrValidate              type=0100
da=00000003:00000000117FE000(3:293593088) sectors=0800 repda=[nVal=-1] data=(len=03000000) 00000000 117FE000
00001000 00000000 00000004 09002003 00000000 00000000 00004E63 00000000 00000000 00010002 00010002 59CC4123
1519D718 59CC4123 169A3588 59CC4123 158FD058 2BD0F913 02008140 00000000 00000000 00000000 00010000 00000000
00000000 00000000 00000000 59CC4123 1519D718 8000 08
```

Reserved Files – Inode File

- Inode number 0
- File data holds fixed size inode records of the rest of the file system
- 1 inode file for each snapshot
- Sparse if independent filesets created



Inode



Inode (cont.)

- Few Corruption Symptoms
  - FSSTRUCT errors in system log
  - admin commands that traverse inodes fail
  - ls: cannot access <file>: Input/output error
- Repair
  - invalid metadata in inode header and trailer - fix values
  - invalid/duplicate disk addresses in inode and indirect blocks - delete disk address
  - invalid metadata in extended attributes - delete bad attribute(s)
  - replica mismatch – copy good replica over bad replicas
  - fatal inode corruptions (invalid checksum, bad indirection levels)
    - delete inode if not reserved file
    - recreate inode for reserved files
      - except inode file, block allocation map file, inode allocation map file

Reserved Files – Block Allocation Map File

- 1 block allocation map file for each storage pool (inode number 1 for system pool)
- Fixed size records (segments) hold allocation state for a subset of blocks in the file system
- 1 bit per sub-block; Cannot be sparse file
- Different bit to sub-block map layouts for FPO and non-FPO file systems

Fixed Size Segment Header	**Disk 0 Header**	**Disk 0 Allocation Index**	**Disk 0 Subset Allocation Bitmap** **(Proportional to disk size)**	**Disk 1 Header**	**Disk 1 Allocation Index**	**Disk 1 Subset Allocation Bitmap**
MagicNumber, Checksum, SegmentLength, FreeSpace	MagicNumber	Linked List / Binary Tree	1 – Free 0 - Allocated	Magic Number	Linked List / Binary Tree	1 – Free 0 - Allocated

Reserved Files – Inode Allocation Map File

- 1 inode allocation map file for file system (inode number 2)
- Fixed size records (segments) hold allocation state of inodes
- 2 bits per inode (for transient states); Sparse like inode file

Fixed Size Segment Header	**Allocation Bitmap for some inodes**
MagicNumber, Checksum, SegmentLength	11 – Free 10 – BeingCreated 01 – ToBeDeleted 00 - Allocated

Allocation Map Corruptions

- Corrupt inode
  - cannot repair file system
- Corrupt indirect block / segments
  - delete bad indirect block / segment
  - fill holes
- Allocation state inconsistency
  - trust inode data over allocation map data
  - in-use in map; free in inode
    - lost block / inode
  - free in map; in-use in inode
    - can cause data loss if not repaired

Reserved Files – Fileset Metadata File

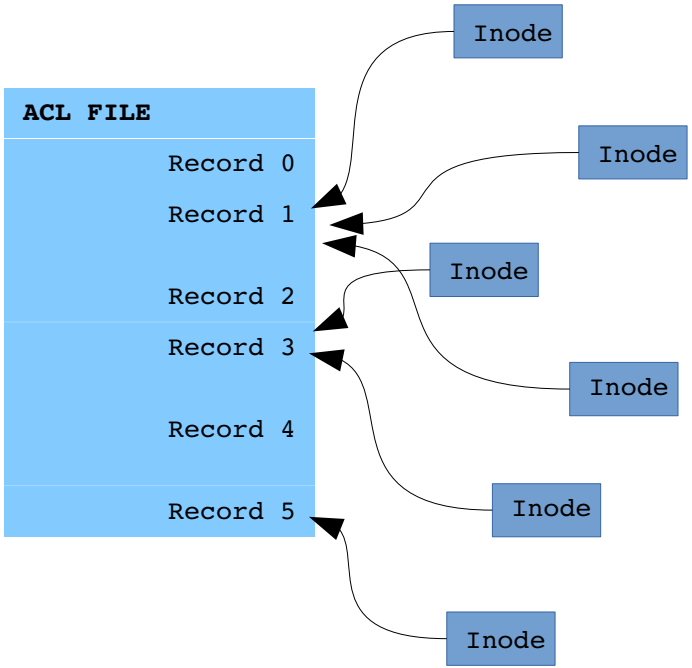
- 1 fileset metadata file per file system
- Each fixed size record contains information about a fileset or fileset snapshots or AFM
- Some important fields
  - fileset name, fileset root directory / junction inode number, parent filesetId
  - inode space index, maxInodes, allocatedInodes
- Few Corruption Symptoms
  - fileset create / delete error
- Repair
  - corrupt fileset metadata file inode – recreate file
  - corrupt fileset record – delete record
  - inconsistent fileset record - scan inodes and directories and re-build record
    - non-redundant fields like fileset name cannot be restored

Reserved Files – Log File

- Multiple log files per file system
  - each node mounting the file system gets a different log file
  - fixed size file
  - reused when file becomes full
- Used to ensure atomic commit of complex updates to file system objects
  - logs of failed nodes are recovered by file system manager node
  - logs recovered on first mount (except for read-only / restricted mounts)
  - Log records are of variable length and chained
- Few Corruption Symptoms
  - log recovery failure -> mount failure
- Repair
  - not easy to repair log records
  - easier to delete and recreate an empty log file

Reserved Files – Access Control List (ACL) File

- 1 ACL file for file system (not copied to snapshots)
- Variable size records store user specified ACLs
  - mmgetacl / mmputacl interface
- Inodes point to the ACL record offset
  - ACL inheritance causes duplication
  - centralized ACL storage has deduplication benefits
  - unused ACL records are garbage collected



ACL Record
Free / In-use Flag
Data Length
Hash Value
Char Data

Reserved Files – ACL file (cont.)

- Few Corruption Symptoms
  - cannot access file
- Repair
  - corrupt ACL record
    - delete record (and all subsequent records in the block)
    - delete inode ACL references to that record
  - corrupt inode ACL reference
    - delete inode ACL reference
    - inode falls back to posix permissions

Reserved Files – Quota Files

- 3 quota files in file system (not copied to snapshots)
  - user, group, fileset
  - created when quota option is enabled

    `mmchfs <fs> -Q yes`
- Records user defined quota limits
  - mmedquota
- Records current quota usage
  - use mmcheckquota to explicity update quota usage
- Repair
  - delete bad quota files
  - subsequent mount will recreate the quota files
  - user sets quotas again

Reserved Files – Policy File

- 1 policy file for file system
  - created if placement policy is installed
  - `mmchpolicy`
- Repair
  - delete bad policy file
  - user installs policies again

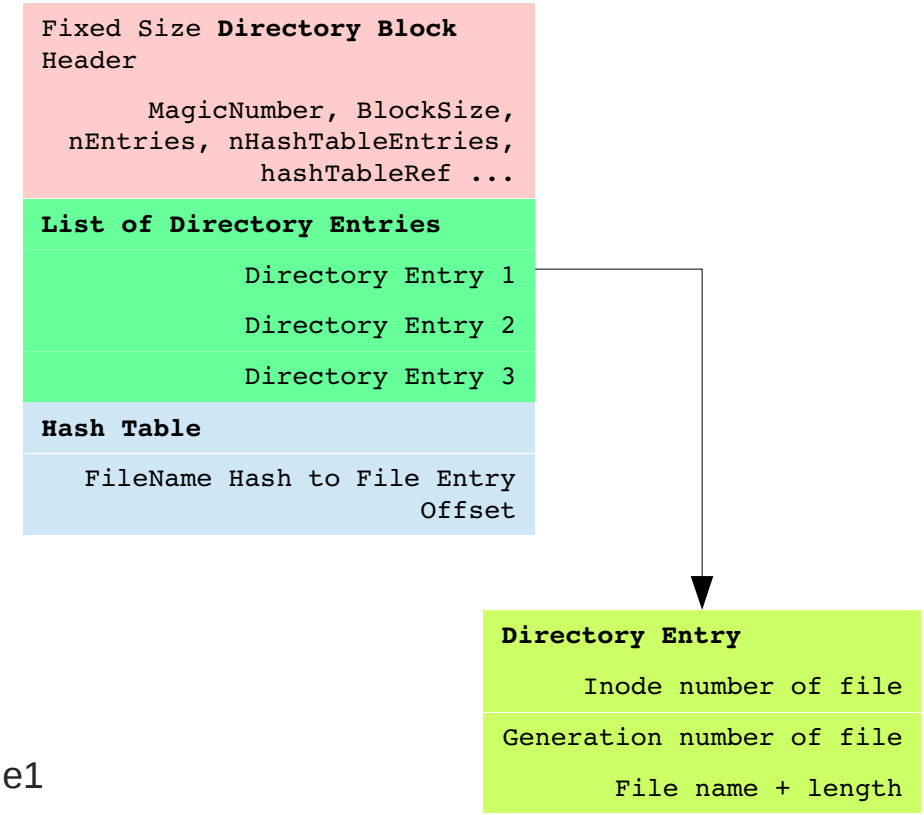
Reserved Files – Allocation Summary File

- 1 allocation summary file per storage pool
  - automatically created as needed
- Basic data needed to respond to a statfs syscall
  - reading all block and inode allocation map segments takes time
  - updated periodically
- Repair
  - delete bad allocation summary file
  - subesquent mount will recreate the allocation summary file

Directory

- 2 layer hash table
  - file name to directory block number
  - file name to directory entry in block
  - overflow block
- '..' entry
  - key to connecting directories
  - found in directory block 0
  - cannot cross fileset boundary
- Few Corruption Symptoms
  - Directory move / delete fails
  - file create / delete fails
  - ls -l

    -????????? ? ? ? ? ? file1



Directory Corruptions

- Corrupt directory block
  - rebuild hash table
  - on failure
    - delete directory if first block / overflow block is corrupt
    - else delete directory block
- Invalid directory entries
  - delete directory entry
- Directory cycles
  - delete all directories in cycle
  - orphan files referenced by the deleted directories
- Ignore corrupt directory in snapshots
  - too expensive to check

FSCK – Behind the Scenes

Online vs Offline FSCK

- Nobody (including FSCK) can recover lost data
  - aim for restoring metadata consistency
- The problem of Global Consistency
  - local consistency – eg. inode checksum
  - global consistency – eg. cross linked blocks
  - difficult when file system is changing under FSCK scan
  - simpler to unmount file system
- Online FSCK
  - can safely identify lost blocks since no file allocator can use such blocks
  - can detect (but not fix) local in-consistencies
  - no directory checks
- Online replica compare and repair
  - `mmrestripefs <fs> -c [--read-only]`

Detecting Corruptions

- Magic numbers and checksums
- Insane values
  - inode.maxMetataReplicas <= MAX_REPLICAS
- Stale values
  - disk address in inode pointing to deleted disk
- Inconsistent values
  - sub-directory’s ‘..’ entry pointing elsewhere

Re-constructing Metadata

- From redundant metadata
  - inode allocation map state from inode status flag
- From derived metadata
  - file size from data block pointers in inode
- By restoring defaults
  - reconstructed fileset is in unlinked state
- By correcting values
  - ‘..’ directory entry cannot have its orphan bit set
- By deleting values
  - delete invalid ACL references

FSCK Phases

- Each repair phase depends on data collected during a previous phase
- Phase 0: Env check and Initialization
  - Command line options
  - Log recovery
  - Allocate node and memory resources
- Phase 1: Reserved files scan & repair
  - single node & single thread
- Phase 2: Inodes and cross linked blocks
  - scan inode file in distributed fashion
  - update in-memory allocation maps
- Phase 3: Directories
  - validate directory blocks and entries
  - check link count & orphan inodes

Distributed FSCK

- Memory distribution
  - uses upto 50% of pagepool and 70% of memory
  - distributes in-memory data structures across all participating nodes
    - in-memory inode and block allocation bitmaps
    - sparse dot-dot array
- Work distribution
  - file system manager node is FSCK master node
  - divide inode range to scan among FSCK worker nodes
  - each worker node assigns multiple threads (default 16) to scan inodes
- Inter-node communication via RPCs
  - phase synchronization
  - send patch records to master node
  - send progress info to master node
  - update distributed in-memory data structures

FSCK Failure Handling

- Startup
  - internal / external mounts
  - version in-compatibility
  - in-sufficient memory
  - critical reserved file health
  - previous fsck not-cleaned up
- Cannot tolerate participating node failure
  - distributed data structures
  - parallel inode scans
- Cleanup
  - node failure handler
  - command socket error detection
  - synchronous cleanup of FSCK on master node
  - asynchronous cleanup of FSCK on worker nodes

FSCK Patch Record

- Patch Record encapsulates a unit of corruption and repair
  - self contained description of corruption
- Patch File can be used to persist list of corruptions during read-only FSCK run
  - apply patch file during subsequent FSCK repair
  - patch file apply is order of minutes – no inode scan required
  - patch file is human readable and can be hand edited
  - enabled via --patch-file & --patch option of offline FSCK
- Patch Queue allows unlimited corruptions to be addresses in a single FSCK run
  - scanning threads create a patch record for every corruption found
  - enqueue into patch queue in FSCK master node
  - patch dequeue thread processes the patches during scan
    - report the corruptions on command line
    - write the patch record to patch file if specified
    - if repair mode, apply the patch record to the corresponding data structure

Cross Linked Blocks

- Disk block is referenced more than once by one or more inodes
- Identifying cross linked blocks
  - start with a zeroed out in-memory block allocation bitmap
  - for every valid disk address in each inode/indirect block, mark corresponding bit in the bitmap
  - if bit is already marked, we have a duplicate address
- Few Corruption Symptoms
  - block contents change without user intervention
  - double deallocation errors
- Repair
  - user data blocks
    - no way to tell who is the rightful owner of the block; delete all block references
  - metadata blocks
    - determine owner using magic number etc.; delete all other block references

Orphans & Snapshots

- Orphan inodes
  - inode allocated but no directory entries pointing to it
  - move to lost+found directory under respective fileset root directory
- Snapshot corruptions
  - corrupt inodes in snapshots are handled the same
  - no checks of directories in snapshots
  - inter-snapshot consistency is not checked
    - eg. missed copy-on-write
    - too expensive to check
  - any error reading snapshot metadata
    - delete all snapshots

Interpreting FSCK output

```
Checking "fs1"
Checking reserved files
Checking inodes
Checking inode map file
Checking directories and files
Checking log files
Checking metadata of filesets
Checking file reference counts
…
                  500224 inodes
                      39   allocated
                       0   repairable
                       0   repaired
                       0   damaged
                       0   deallocated
                       0   orphaned
                       0   attached

                13107200 subblocks
                  142696   allocated
                       0   unreferenced
                    4209 addresses
                       0   reserved file holes found
                       0   reserved file holes repaired
File system is clean.
```

```
Unsuccessful FSCK ends with -

File system contains unrepaired damage.
Exit status <err>:<code>:<status>

err is any file system error which caused FSCK to abort
code is meant for internal debugging purpose
status can be –

2: INTERRUPTED – premature exit
8: UNFIXED – user declined to fix problems
16: UNFIXABLE – some problems could not be fixed
32: BENIGN – some non-critical problems were not fixed
```

How long does FSCK take?

- Umm...
- Many variables
  - size of file system / number of blocks to scan
    - system pool with data & metadata
    - nature of corruptions (cross linked blocks require additional inode scan)
  - distribution of files and directories among inode space
    - files of varying sizes may cause some nodes to take longer to finish the scan
    - ratio of directories to files in the file system can change directory scan time
  - I/O throughput
    - uneven disk / network throughput
    - other workloads in the cluster
  - multiple inode scans may be required depending on available pagepool memory
- But ...can estimate using internal dump from running file system and mmdf output

Best Practices

Should I run Offline FSCK?

- FSSTRUCT errors: not all of them are serious
  - only one replica reported
    - `mmrestripefs <fs> -c`
  - snapshot related
    - `mmdelsnapshot`
    - `tsdbfs <fs> patch desc nsgsnapitems 0`
  - use tsdbfs to verify corruption is on disk
  - use mmfileid to check for cross linked blocks

    `mmfileid <fs> -d :<disk address>`
- 'ls -l' errors: -????????? ? ? ? ? ?
  - tsdbfs can help delete the file
- Too many corruptions: permanent disk loss
  - online fsck can show (but not fix) inode problems
  - faster to recreate file system

Read-only Offline FSCK before repair

- Know extent of corruptions
- 'tsfindinode'
  - match inode numbers to file names
  - copy out data that may be deleted by fsck
  - restore from backup after fsck repair
- --patch-file & --patch options

```
mmfsck <fs> -vn --patch-file <file-name> 2>&1 | tee fsckn.out
mmfsck <fs> -v --patch –patch-file <file-name> 2>&1 | tee fsckp.out
```

  - patch file is editable
  - undo repair using patch file
  - some reserved file corruptions require full offline fsck

Faster Offline FSCK

- Increase FSCK threads

```
mmfsck <fs> --threads 128
```

- Continue fsck with multiple inode scan passes? **N**
  - increase FSCK usage of pagepool

```
mmdsh -N all mmfsadm test fsck pctmem 75
```

  - increase pagepool memory
  - increase nodes participating in FSCK
- Prefer NSD server nodes as FSCK workers

```
mmfsck <fs> -N node1,node2
```

- Suppress replica compare if log recovery fails

```
mmfsck <fs> -xc
```

- Prefer metadata only replica compare

```
mmfsck <fs> -m
```

Faster Offline FSCK - cont

- Skip directory cycle check

```
mmfsck <fs> -xsc
```

- Repair inode and allocation maps only

```
mmfsck <fs> --skip-directory-check
mmfsck <fs> --skip-inode-check
```

- Repair log files only

```
mmfsck <fs> -xk
```

- Delete old snapshots
- Preferable not to run with other workloads
- Safe to abort FSCK (ctrl-c)

General Best Practices

- Do not run FSCK with down disks
- Always save FSCK outputs with time-stamps
- Good idea to attach – tsdbfs <fs> desc – output
- Read only FSCK after repair
  - for short fsck runs
- Monitor system logs for FSSTRUCT errors
  - mmhealth
  - mmaddcallback --event fsstruct
  - assertOnStructureError = no
- Offline FSCK before upgrade

References

- Product documentation -
  - “Checking and repairing a file system”
  - “The mmfsck command”
  - man mmfsck
- IBM Developer Works
  - Forum: General Parallel File System (GPFS)
  - IBM Spectrum Scale Wiki

Thank You

karthik.iyer@in.ibm.com