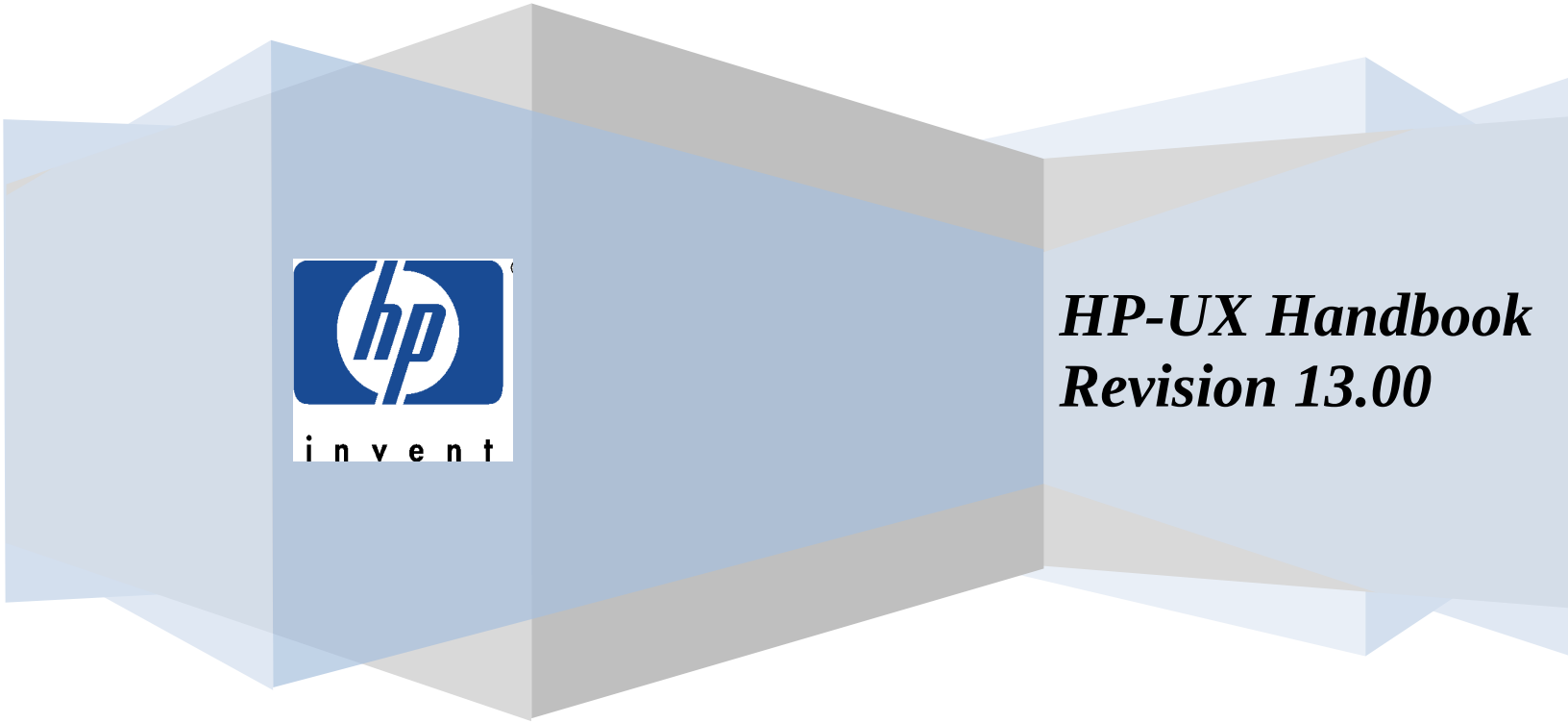
Chapter 14

Journaled File System (JFS)



HP-UX Handbook
Revision 13.00

TERMS OF USE AND LEGAL RESTRICTIONS FOR THE HP-UX RECOVERY HANDBOOK

ATTENTION: PLEASE READ THESE TERMS CAREFULLY BEFORE USING THE HP-UX HANDBOOK. USING THESE MATERIALS INDICATES THAT YOU ACCEPT THESE TERMS. IF YOU DO NOT ACCEPT THESE TERMS, DO NOT USE THE HP-UX HANDBOOK.

THE HP-UX HANDBOOK HAS BEEN COMPILED FROM THE NOTES OF HP ENGINEERS AND CONTAINS HP CONFIDENTIAL INFORMATION.

THE HP-UX HANDBOOK IS NOT A PRODUCT QUALITY DOCUMENT AND IS NOT NECESSARILY MAINTAINED OR UP TO DATE. THE HP-UX HANDBOOK IS HERE MADE AVAILABLE TO HP CONTRACT CUSTOMERS FOR THEIR INTERNAL USE ONLY AND ON THE CONDITION THAT NEITHER THE HP-UX HANDBOOK NOR ANY OF THE MATERIALS IT CONTAINS IS PASSED ON TO ANY THIRD PARTY.

Use of the HP-UX Handbook: Hewlett-Packard Company ("HP") authorizes you to view and download the HP-UX Handbook only for internal use by you, a valued HP Contract Customer, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials. You may not modify the HP-UX Handbook in any way or publicly display, perform, or distribute or otherwise use them for any public or purpose outside your own business. The materials comprising the HP-UX Handbook are copyrighted and any unauthorized use of these materials may violate copyright, trademark, and other laws. If you breach any of these Terms, your authorization to use the HP-UX Handbook automatically terminates and you must immediately destroy any downloaded or printed materials.

Links To Other Web Sites: Links to third party Web sites provided by the HP-UX Handbook are provided solely as a convenience to you. If you use these links, you will leave this Site. HP has not reviewed all of these third party sites and does not control and is not responsible for any of these sites or their content. Thus, HP does not endorse or make any representations about them, or any information, software or other products or materials found there, or any results that may be obtained from using them. If you decide to access any of the third party sites linked to this Site, you do this entirely at your own risk.

Disclaimer: THE HP-UX HANDBOOK IS PROVIDED "AS IS" WITHOUT ANY WARRANTIES OF ANY KIND INCLUDING WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF INTELLECTUAL PROPERTY. HP further does not warrant the accuracy and completeness of the materials in the HP-UX Handbook. HP may make changes to the HP-UX Handbook at any time without notice. The HP-UX Handbook may be out of date, and HP makes no commitment to update the HP-UX Handbook. Information in the HP-UX Handbook may refer to products, programs or services that are not available in your country. Consult your local HP business contact for information regarding the products, programs and services that may be available to you.

Limitation of Liability: IN NO EVENT WILL HP, ITS SUPPLIERS, OR OTHER ANY THIRD PARTIES MENTIONED IN THE HP-UX HANDBOOK BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, THOSE RESULTING FROM LOST PROFITS, LOST DATA OR BUSINESS INTERRUPTION) ARISING OUT OF THE USE, INABILITY TO USE, OR THE RESULTS OF USE OF THE HP-UX HANDBOOK, WHETHER BASED ON WARRANTY, CONTRACT, TORT OR ANY OTHER LEGAL THEORY AND WHETHER OR NOT ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS DOES NOT APPLY IN CASE OF INTENT OR IF LIABILITY IS LEGALLY STIPULATED. IF YOUR USE OF THE HP-UX HANDBOOK RESULTS IN THE NEED FOR SERVICING, REPAIR OR CORRECTION OF EQUIPMENT OR DATA, YOU ASSUME ALL COSTS THEREOF.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Applicable Laws: These Terms will be governed by and construed in accordance with the laws of the State of California, without giving effect to any principles of conflicts of laws.

General: HP may revise these Terms at any time by updating this posting. ***Revised Oct 2013***

FEEDBACK or QUESTIONS: please email essam.ackleh@hp.com
(please use subject syntax: *HP-UX Handbook v13.00 Chapter <YY> - <Feedback Title>*

TABLE OF CONTENTS

Introduction:

The VERITAS File System, (or VxFS, called JFS and OnlineJFS in HP-UX), is an extent-based file system. It was originally developed by VERITAS Software.[1] Through an OEM agreement, VxFS is used as the primary files system of the HP-UX operating system. With on-line defragmentation and resize support turned on via license, it is known as *OnlineJFS*.
The current revision is 5.1

A file system is simply a method for storing and organizing computer files and the data they contain to make it easy to find and access them. More formally, a file system is a set of abstract data types (such as metadata) that are implemented for the storage, hierarchical organization, manipulation, navigation, access, and retrieval of data.

Veritas File System (VxFS) was the first commercial journaling file system.

With journaling, metadata changes are first written to a log (or journal) then to disk. Since changes do not need to be to be written in multiple places, throughput is much faster as the metadata is written asynchronously.

VxFS is also an extent-based, intent logging file system. VxFS is designed for use in operating environments that require high performance and availability and deal with large amounts of data.

VxFS major components include:

- Logging
- Extents
- File system disk layouts

Logging

A key aspect of any file system is how to recover if a system crash occurs. Earlier methods required a time-consuming scan of the entire file system. A better solution is the method logging (or journaling) the metadata of files.

VxFS logs new attribute information into a reserved area of the file system, whenever file system changes occur. The file system writes the actual data to disk only after the write of the metadata to the log is complete. If and when a system crash occurs, the system recovery code analyzes the metadata log and try to clean up only those files. Without logging, a file system check (fsck) must look at all of the metadata.

Intent logging minimizes system downtime after abnormal shutdowns by logging file system

transactions. When the system is halted unexpectedly, this log can be replayed and outstanding transactions completed. The check and repair time for file systems can be reduced to a few seconds, regardless of the file system size. By default, VxFS file systems log file transactions before they are committed to disk, reducing time spent checking and repairing file systems after the system is halted unexpectedly.

Extents

An extent is a contiguous area of storage in a computer file system, reserved for a file. When starting to write to a file, a whole extent is allocated. When writing to the file again, the data continues where the previous write left off. This reduces or eliminates file fragmentation.

Since VxFS is an extent-based file system, addressing is done through extents (which can consist of multiple blocks) rather than in single blocks segments. Extents can therefore enhance file system throughput.

File system disk layouts

The disk layout is the way file system information is stored on disk. On VxFS, several disk layout versions, numbered 1 through 7, were created to support various new features and specific UNIX environments. Currently, only the Version 4, 5, 6, and 7 disk layouts can be created and mounted.

The on-disk layout of VxFS is versioned and upgradeable while the file system is mounted. This file system has gone through seven versions.

- Version 2 added support for filesets, dynamic inode allocation and ACLs. Layouts 1-3 stopped being supported in VxFS 4.0.

- Version 4 added support for storage checkpoints and for Veritas Cluster File System. Version 4 was released in VxFS 3.2.1. Layout version 4 is no longer supported under VxFS 5.1

- Version 5 started support for file systems up to 32 terabytes (2^{45} bytes) in size.
- Individual files can be up to 2 terabytes in size. Version 5 was introduced in VxFS 3.5 and is no longer supported under VxFS 5.1

- Version 6 added support for file systems and files up to 8 exabytes (2^{63} bytes) in size. Version 6 also introduced support for named streams/resource forks, for multiple underlying volumes, and for file change logs. Version 6 was introduced in VxFS 4.0.

- Version 7 extends support for multiple volumes to permit Dynamic Storage Tiering. Dynamic Storage Tiering allows root users to move files among different volumes, allocate files to different volumes at file creation time based on policy, and independently recover volumes, without altering the namespace of the file system. Version 7 was introduced in VxFS 5.0

VxFS 5.X enables you to mount the following file system disk layouts:

- Disk Layout Version 4
- Disk Layout Version 5
- Disk Layout Version 6
- Disk Layout Version 7

The default layout for VxFS 5.0 is Disk Layout Version 7. Any new file system created using the VxFS 5.0
mkfs command has Disk Layout Version 7, unless explicitly specified

- **Do not upgrade / and /stand file systems to Disk Layout Version 6 or Disk Layout Version 7.**

The HP-UX boot loader does not support the Disk Layout Version 6 or Disk Layout 7.

Veritas File System features

- **Extent-based allocation**
- Extents allow disk I/O to take place in units of multiple blocks if storage is
- allocated in consecutive blocks.
- **Extent attributes**
- Extent attributes are the extent allocation policies associated with a file.
- **Fast file system recovery**
- VxFS provides fast recovery of a file system from system failure.

- **Extended mount options**
  - The VxFS file system supports extended mount options to specify enhanced data integrity modes, enhanced performance modes, temporary file system modes, improved synchronous writes, and large file sizes.
- **Enhanced performance mode**
  - VxFS provides mount options to improve performance.
- **Large files and file systems support**

  - VxFS supports files larger than two terabytes and large file systems up to 256 terabytes.
- **Online backup**
  - VxFS provides online data backup using the snapshot feature.
- **Quotas**
  - VxFS supports quotas, which allocate per-user and per-group quotas and limit the use of two principal resources: files and data blocks.
- **Cluster File System**
  - Clustered file systems are an extension of VxFS that support concurrent direct media access from multiple systems.
- **Improved database performance**
- **Storage Checkpoints**
  - Backup and restore applications can leverage Storage Checkpoint, a disk- and I/O-efficient copying technology for creating periodic frozen images of a file system.
- **Cross-platform data sharing**
  - Cross-platform data sharing allows data to be serially shared among heterogeneous systems where each system has direct access to the physical devices that hold the data.
- **File Change Log**
  - The VxFS File Change Log tracks changes to files and directories in a file system.

- **Multi-volume support**
  - The multi-volume support feature allows several volumes to be represented by a single logical object.

- **Dynamic Storage Tiering**
  - The Dynamic Storage Tiering (DST) option allows you to configure policies that automatically relocate files from one volume to another, or relocate files by running file relocation commands, which can improve performance for applications that access specific types of files.

Note: VxFS supports all HFS file system features and facilities except for the linking, removing, or renaming of "." and ".." directory entries. Such operations may disrupt file system sanity.

Upgrading VxFS

• **Do not upgrade the operating system and the Veritas products simultaneously**.
HP recommends that clients first upgrade the operating system and later upgrade the Veritas products.

NOTE: In VxFS 4.1 the default disk layout was version 6. File system commands like df, bdf, fstyp and mount that use the API functions statfsdev (), fstatfsdev (), statvfsdev (), or fsstatvfsdev () failed because the libc library did not recognize the new Disk Layout Version 6. Applications using statvfsdev() had to be linked with the new NCF library to recognize Disk Layout Version 6.

In 5.X, the default disk layout is version 7, so the file system commands delivered with this product have been relinked to support Disk Layout Version 7. However, third-party Applications must be relinked again in order to recognize Disk Layout Version 7.

The upgrade process:

1. To determine the volume names, enter the following:
   a. # bdf
2. To check for DLV 2 or 3 partitions present in the system (since these file systems cannot be mounted
3. after upgrading to VxFS 5.0), enter the following:
   a. # fstyp -v <partition_name>
4. To upgrade file systems with DLV 2 or 3, before installing 5.0, complete the following steps (You cannot upgrade from DLV 2 to DLV 4 directly. DLV4 is the lowest disk layout that is supported by VxFS 5.0.):
   a. To upgrade a mounted VxFS file system from DLV 2 to DLV 3, enter the following:
   b. # vxupgrade -n 3 <dir_name>
   c. To upgrade a mounted VxFS file system from DLV 3 to DLV 4, enter the following:
   d. # vxupgrade -n 4 <dir_name>

5. **Upgrade to the current version of HP-UX 11i** , refer to the OS upgrade and Install guide .

6. **Install VxFS 5. X**

7. If there are any VxFS file systems, with disk layout 2 or 3 or non-VxFS file systems on your system, after upgrading to VxFS 5.X, that must be converted to VxFS.

   To upgrade to DLV 7 and convert an HFS file system to VxFS, enter the following:
   # /opt/VRTS/bin/vxfsconvert <vol_name>

   The *vxfsconvert* command works with unmounted file systems. To upgrade mounted file

system use the *vxupgrade* command as described in step 8. You must run **fsck** after *vxfsconvert* as *vxfsconvert* does not create all metadata files.

8. To run the VxFS-specific full fsck on the converted file system, enter the following:
   # fsck -F vxfs -y -o full <vol_name>

   During pass 4, fsck displays several error messages that require a yes response to complete the conversion process.
9. VxFS file systems with disk layout 4 can still be mounted with VxFS 5.0. However, to upgrade to disk layout versions 5, 6, or 7, use *vxupgrade* as follows:

   a. To upgrade a mounted VxFS file system from disk layout 4 to disk layout 5
      ***# vxupgrade -n 5* <mount_point>**

   b. To upgrade a mounted VxFS file system from disk layout 4 to disk layout 6
      ***# vxupgrade -n 5* <mount_point>**
      ***# vxupgrade -n 6* <mount_point>**

   c. To upgrade a mounted VxFS file system from disk layout 4 to disk layout 7
      ***# vxupgrade -n 5* <mount_point>**
      # vxupgrade -n 6 <mount_point>
      # vxupgrade -n 7 <mount_point>
10. To verify the conversion, enter the following:
    # fstyp -v <vol_name>

Access Control Lists

An Access Control List (ACL) stores a series of entries that identify specific users or groups and their access privileges for a directory or file. A file may have its own ACL or may share an ACL with other files. ACLs have the advantage of specifying detailed access permissions for multiple users and groups.

File systems created with the Version 5, 6, or 7 disk layouts, up to 1024 ACL entries can be specified. ACLs are also supported on cluster file systems. See the getacl(1) and setacl(1) manual pages.

NOTE: only available for disk layout 4 and above.

1) Display an access control list (ACL) for files (JFS File systems only)

 # getacl <file>

 # file: filename
owner: uid
group: gid
user::perm
user:uid:perm

group::perm
group:gid:perm
class:perm
other:perm
default:user::perm
default:user:uid:perm
default:group::perm
default:group:gid:perm
default:class:perm
default:other:perm

2) setacl - modify access control lists (ACLs) for files (JFS File systems only)

Example:
To add one ACL entry to file "testfile", giving user root read permission only, type:

 # setacl -m user:root:r - - testfile

 # getacl testfile

 # file: testfile
 # owner: root
 # group: sys
user::rw-
user:root:r--
group::r--
class:r--
other:r—

You have also the possibility to define the ACL List within a testfile.acl to convert the entries to the testfile.

Quotas

VxFS supports quotas, which allocate per-user quotas and limit the use of two principal resources: files and data blocks. You can assign quotas for each of these resources. Each quota consists of two limits for each resource: hard limit and soft limit. The soft limit can be exceeded for a given time limit. This time limit is per default 7 days and can be changed via *edquota –t.*

When a user exceeds the soft limit a warning is displayed. The hard limit is an absolute limit, which cannot be exceeded, so the soft limit should be lower than the hard limit.
To use quotas a file quotas must exist in the root directory of the file system. The quotas are set by using *edquota* which opens an editor to set the soft and hard limits. The quotas can be turned on either by the command *quotaon* or by the mount options (-o quota)

Creating the quotas file under the root directory of the file system

touch /<moutpoint>/quotas (must be owned by root)

setting quotalimits for a via edquota (can be used like vi)user:

#edquota <username>
e.g.
fs /home blocks (soft = 10, hard = 20) inodes (soft = 10, hard = 20)
fs / blocks (soft = 10, hard = 20) inodes (soft = 10, hard = 20)

edit the limits

#quotaon

#quotacheck <mountpoint>

#quota –v <username> to display the quotas

Example:
quota -v darrit
Disk quotas for darrit (uid 103):
File system usage quota limit timeleft files quota limit timeleft
/home 7 10 20 7 10 20
The quotas can also be turned on by the mount options
#mount –F vxfs –o quota /dev/vgXX/lvolY/ /mountpoint

To turn off quota use
quotaoff

Online backup

VxFS provides online data backup using the snapshot feature. An image of a mounted file system instantly becomes an exact read-only copy of the file system at a specific point in time. The original file system is called the snapped file system, the copy is called the snapshot.

When changes are made to the snapped file system, the old data is copied to the snapshot. When the snapshot is read, data that has not changed is read from the snapped file system, changed data is read from the snapshot.

Backups require one of the following methods:

- Copying selected files from the snapshot file system (using *find* and *cpio)*
- Backing up the entire file system (using *fscat*)
- Initiating a full or incremental backup (using *vxdump*)

Support for databases

Databases are usually created on file systems to simplify backup, copying, and moving tasks and are slower compared to databases on raw disks.

Using Quick I/O for Databases feature with VxFS lets systems retain the benefits of having a database on a file system without sacrificing performance.

Veritas Quick I/O creates regular, pre-allocated files to use as character devices. Databases can be created on the character devices to achieve the same performance as databases created on raw disks. Treating regular VxFS files as raw devices has the following advantages for databases:

Commercial database servers such as Oracle Server can issue kernel supported asynchronous I/O calls (through the asyncdsk or Posix AIO interface) on these pseudo devices but not on regular files.

Cluster file systems

Veritas Storage Foundation Cluster File System (SFCFS) allows clustered severs to mount and use a file system simultaneously as if all applications using the file system were running on the same server. The Veritas Volume Manager cluster functionality (CVM) makes logical volumes and raw device applications accessible through a cluster.

Beginning with SFCFS 5.0, SFCFS uses a symmetric architecture in which all nodes in the cluster can simultaneously function as metadata severs. SFCFS still as some remnants of the old master/slave or primary/secondary concept. The first server to mount each cluster file system becomes its primary; all other nodes in the cluster become secondaries. Applications access the user data in files directly from the server on which they are running. Each SFCFS node has its own intent log. File system operations, such as allocating or deleting files, can originate from any node in the cluster.

NOTE: Installing VxFS and enabling the cluster feature does not create a cluster file system configuration. HP Serviceguard Storage Management environments require HP Serviceguard for file system clustering.

To be a cluster mount, a file system must be mounted using the mount -o cluster option. File systems mounted without the -o cluster option are termed local mounts.

Cross-platform data sharing
Cross-platform data sharing (CDS) allows data to be serially shared among heterogeneous systems where each system has direct access to the physical devices that hold the data.

NOTE: This feature can be used only in conjunction with Veritas Volume Manager

(VxVM).

File Change Log

The VxFS File Change Log (FCL) tracks changes to files and directories in a file system. The File Change Log can be used by applications such as backup products, webcrawlers, search and indexing engines, and replication software that typically scan an entire file system searching for modifications since a previous scan.

NOTE: FCL functionality is a separately licensed feature.

Multi-volume support

The multi-volume support (MVS) feature allows several volumes to be represented by a single logical object. All I/O to and from an underlying logical volume is directed by way of volume sets.

This feature can be used only in conjunction with VxVM.

NOTE: MVS functionality is a separately licensed feature.

Dynamic Storage Tiering

The Dynamic Storage Tiering (DST) option is built on multi-volume support technology. Using DST, you can map more than one volume to a single file system. You can then configure policies that automatically relocate files from one volume to another, or relocate files by running file relocation commands. Having multiple volumes lets you determine where files are located, which can improve performance for applications that access specific types of files.

NOTE:
DST functionality is a separately licensed feature and is available with the VRTSfppm package.

Veritas File System performance enhancements

Traditional file systems employ block-based allocation schemes that provide adequate random access and latency for small files, but which limit throughput for larger files. As a result, they are less than optimal for commercial environments.

VxFS addresses this file system performance issue through an alternative allocation method and increased user control over allocation, I/O, and caching policies.

VxFS provides the following performance enhancements:

- Data synchronous I/O
- Direct I/O and discovered direct I/O
- Support for files and file systems up to 256 terabytes
- Support for files up to 2 terabytes
- Enhanced I/O performance
- Caching advisories
- Enhanced directory features
- Explicit file alignment, extent size, and pre-allocation controls
- Tunable I/O parameters
- Tunable indirect data extent size
- Integration with VxVM
- Support for large directories

Note: VxFS reduces the file lookup time in directories with an extremely large number of files.

Improving I/O using vxtunefs

This command can be used to print or set tunable I/O parameters of mounted file systems. The I/O parameters are:

- Parameters describing the I/O properties of the underlying device
- Parameters to indicate when to treat an I/O as direct I/O
- Parameters to control the extend allocation policy for the specific file systems

It works on a list of mount points specified on the command line or all the mounted file systems listed in the tunefstab file, which is in /etc/vx/tunefstab. The change of the parameters takes place immediately.
To print the values, type the following command:
vxtunefs -p *mount_point*

The following is an example tunefstab file:

/dev/vx/dsk/userdg/netbackup
read_pref_io=128k,write_pref_io=128k,read_nstream=4,write_nstream=4
/dev/vx/dsk/userdg/metasave
read_pref_io=128k,write_pref_io=128k,read_nstream=4,write_nstream=4
/dev/vx/dsk/userdg/solbuild
read_pref_io=64k,write_pref_io=64k,read_nstream=4,write_nstream=4
/dev/vx/dsk/userdg/solrelease
read_pref_io=64k,write_pref_io=64k,read_nstream=4,write_nstream=4
/dev/vx/dsk/userdg/solpatch
read_pref_io=128k,write_pref_io=128k,read_nstream=4,write_nstream=4

Tunable VxFS I/O parameters

1) print the tuning parameters for all file systems specified on the command line:

   # vxtunefs –p [{<mountpoint>|<block-special>}]

2) setting of new parameters for the file system:

   # vxtunefs –s –o <parameter=value> [{<mountpoint>|<block-special>}]

   -s = sets the parameter

Parameters

read_pref_io

The preferred read request size. The file system uses this in conjunction with the read_nstream value to determine how much data to read ahead.

The default value is 64K.

write_pref_io

The preferred write request size. The file system uses this in conjunction with the write_nstream value to determine how to do flush behind on writes.

The default value is 64K.

read_nstream

The number of parallel read requests of size read_pref_io to have outstanding at one time. The file system uses the product of read_nstream multiplied by read_pref_io to determine its read ahead size.

The default value for read_nstream is 1.

write_nstream

The number of parallel write requests of size write_pref_io to have outstanding at one time. The file system uses the product of write_nstream multiplied by write_pref_io to determine when to do flush behind on writes.

The default value for write_nstream is 1.

discovered_direct_iosz

Any file I/O requests larger than discovered_direct_iosz are handled as discovered direct I/O. A discovered direct I/O is unbuffered similar to direct I/O, but it does not require a synchronous commit of the inode when the file is extended or blocks are allocated. For larger I/O requests, the CPU time for copying the data into the page cache and the cost of using memory to buffer the I/O data becomes more expensive than the cost of doing the disk I/O. For these I/O requests, using discovered direct I/O is more efficient than regular I/O.

The default value of this parameter is 256K.

fcl_keeptime

Specifies the minimum amount of time, in seconds, that the VxFS File Change Log (FCL) keeps records in the log. When the oldest 8K block of FCL records have been kept longer than the value of fcl_keeptime, they are purged from the FCL and the extents nearest to the beginning of the FCL file are freed. This process is referred to as "punching a hole." Holes are punched in the FCL file in 8K chunks.

If the fcl_maxalloc parameter is set, records are purged from the FCL if the amount of space allocated to the FCL exceeds fcl_maxalloc, even if the elapsed time the records have been in the log is less than the value of fcl_keeptime. If the file system runs out of space before fcl_keeptime is reached, the FCL is deactivated.

Either or both of the fcl_keeptime or fcl_maxalloc parameters must be set before the File Change Log can be activated. fcl_keeptime does not apply to disk layout Versions 1 through 5.

fcl_maxalloc

Specifies the maximum amount of space that can be allocated to the VxFS File Change Log (FCL). The FCL file is a sparse file that grows as changes occur in the file system. When the space allocated to the FCL file reaches the fcl_maxalloc value, the oldest FCL records are purged from the FCL and the extents nearest to the beginning of the FCL file are freed.

This process is referred to as "punching a hole." Holes are punched in the FCL file in 8K chunks. If the file system runs out of space before fcl_maxalloc is reached, the FCL is deactivated.

Either or both of the fcl_maxalloc or fcl_keeptime parameters must be set before the File Change Log can be activated. fcl_maxalloc does not apply to disk lay out Versions 1 through 5.

fcl_winterval

Specifies the time, in seconds, that must elapse before the VxFS File Change Log (FCL) records a data overwrite, data extending write, or data truncate for a file. The ability to limit the number of repetitive FCL records for continuous writes to the same file is important for file system performance and for applications processing the FCL.

The fcl_winterval is best set to an interval less than the shortest interval between reads of the FCL by any application. This way all applications using the FCL can be assured of finding at least one FCL record for any file experiencing continuous data changes.

fcl_winterval is enforced for all files in the file system. Each file maintains its own time stamps, and the elapsed time between FCL records is per file. This elapsed time can be overridden using the VxFS FCL sync public API. See the vxfs_fcl_sync(3) manual page.
fcl_winterval does not apply to disk layout Versions 1 through 5.

hsm_write_ prealloc

For a file managed by a hierarchical storage management (HSM) application, sm_write_prealloc preallocates disk blocks before data is migrated back into the file system.

An HSM application usually migrates the data back through a series of writes to the file, each of which allocates a few blocks. By setting hsm_write_ prealloc (hsm_write_ prealloc=1), a sufficient number of disk blocks are allocated on the first write to the empty file so
that no disk block allocation is required for subsequent writes. This improves the write performance during migration.

The hsm_write_ prealloc parameter is implemented outside of the DMAPI specification, and its

usage has limitations depending on how the space within an HSM-controlled file is managed.

NOTE: It is advisable to use hsm_write_ prealloc only when recommended by the HSM application controlling the file system.

max_seqio_extent_size

Changes the default initial extent size. VxFS determines, based on the first write to a new file, the size of the first extent to be allocated to the file. Normally the first extent is the smallest power of 2 that is larger than the size of the first write. If that power of 2 is less than 8K, the first extent allocated is 8K. After the initial extent, the file system increases the size of subsequent extents with each allocation.

initial_extent_size

Since most applications write to files using a buffer size of 8K or less, the increasing extents start doubling from a small initial extent. initial_extent_size can change the default initial extent size to be larger, so the doubling policy starts from a much larger initial size and the file system does not allocate a set of small extents at the start of file.

Use this parameter only on file systems that have a very large average file size. On these file systems it results in fewer extents per file and less fragmentation.

initial_extent_size is measured in file system blocks.

inode_aging_count

Specifies the maximum number of inodes to place on an inode aging list. Inode aging is used in conjunction with file system Storage Checkpoints to allow quick restoration of large, recently deleted files. The aging list is maintained in first-in-first-out (fifo) order up to maximum number of inodes specified by inode_aging_count. As newer inodes are placed on the list, older inodes are removed to complete their aging process. For best performance, it is advisable to age only a limited number of larger files before completion of the removal process.
The default maximum number of inodes to age is 2048.

max_buf_data_size

The maximum buffer size allocated for file data; either 8K bytes or 64K bytes. Use the larger value for workloads where large reads/writes are performed sequentially. Use the smaller value

on workloads where the I/O is random or is done in small chunks.

8K bytes is the default value.

inode_aging_size

Specifies the minimum size to qualify a deleted inode for inode aging. Inode aging is used in conjunction with file system Storage Checkpoints to allow quick restoration of large, recently deleted files. For best performance, it is advisable to age only a limited number of larger files before completion of the removal process.

Setting the size too low can push larger file inodes out of the aging queue to make room for newly removed smaller file inodes.

max_direct_iosz

The maximum size of a direct I/O request that are issued by the file system. If a larger I/O request comes in, then it is broken up into *max_direct_iosz* chunks. This parameter defines how much memory an I/O request can lock at once.

NOTE: This parameter should not be set to more than 20 percent of memory.

max_diskq

Limits the maximum disk queue generated by a single file. When the file system is flushing data for a file and the number of buffers being flushed exceeds max_diskq, processes are blocked until the amount of data being flushed decreases. Although this does not limit the actual disk queue, it prevents flushing processes from making the system unresponsive.

The default value is 1 MB.

max_seqio_extent_size

Increases or decreases the maximum size of an extent. When the file system is following its default allocation policy for sequential writes to a file, it allocates an initial extent which is large enough for the first write to the file. When additional extents are allocated, they are progressively larger because the algorithm tries to double the size of the file with each new extent. As such, each extent can hold several writes worth of data. This is done to reduce the total number of extents in anticipation of continued sequential writes.

When the file stops being written, any unused space is freed for other files to use.
Normally this allocation stops increasing the size of extents at 2048 blocks which prevents one file from holding too much unused space.

max_seqio_extent_size is measured in file system blocks. The default and minimum value of max_seqio_extent_size is 2048.

default_indir_ size

On VxFS, files can have up to ten direct extents of variable size stored in the inode. After these extents are used up, the file must use indirect extents which are a fixed size that is set when the file first uses indirect extents.

These indirect extents are 8K by default. The file system does not use larger indirect extents because it must fail a write and return ENOSPC if there are no extents available that are the indirect extent size.

For file systems with many large files, the 8K indirect extent size is too small. The files that get into indirect extents use many smaller extents instead of a few larger ones. By using this parameter, the default indirect extent size can be increased so large that files in indirects use fewer larger extents.

NOTE: **The tunable default_indir_ size should be used carefully**. If it is set too large, then writes fail when they are unable to allocate extents of the indirect extent size to a file. In general, the fewer and the larger the files on a file system, the larger the default_indir_ size can be set.

This parameter should generally be set to some multiple of the read_pref_io parameter. default_indir_ size is not applicable on Version 4 disk layouts.

qio_cache_enable

Enables or disables caching on Quick I/O files. The default behavior is to disable caching. To enable caching, set *qio_cache_enable* to 1. On systems with large memories, the database cannot always use all of the memory as a cache.

By enabling file system caching as a second level cache, performance may be improved. If the database is performing sequential scans of tables, the scans may run faster by enabling file system caching so the file system performs aggressive read-ahead on the files.

read_ahead

The default for all VxFS read operations is to perform sequential read ahead. You can specify the read_ahead cache advisory to implement the VxFS enhanced read ahead functionality. This allows read aheads to detect more elaborate patterns, such as increasing or decreasing read offsets or multithreaded file accesses, in addition to simple sequential reads. You can specify the following values for read_ahead:

0—Disables read ahead functionality
1—Retains traditional sequential read ahead behavior
2—Enables enhanced read ahead for all reads

The default is 1

vx_era_nthreads

VxFS detects only sequential patterns, read_ahead detects patterns on a per-thread basis, up to a maximum determined by vx_era_nthreads parameter.

The default number of threads is 5, but you can change the default value by setting the vx_era_nthreads parameter in the /etc/system configuration file.

write_throttle

The write_throttle parameter is useful in special situations where a computer system has a combination of a large amount of memory and slow storage devices.

In this configuration, sync operations, such as fsync(),may take long enough to complete that a system appears to hang. This behavior occurs because the file system is creating dirty buffers (in-memory updates) faster than they can be asynchronously flushed to disk without slowing system performance.

Lowering the value of write_throttle limits the number of dirty buffers per file that a file system generates before flushing the buffers to disk. After the number of dirty buffers for a file reaches the write_throttle threshold, the file system starts flushing buffers to disk even if free memory is still available.

The default value of write_throttle is zero, which puts no limit on the number of dirty buffers per file.

If non-zero, VxFS limits the number of dirty buffers per file to write_throttle buffers. The default value typically generates a large number of dirty buffers, but maintains fast user writes.

Depending on the speed of the storage device, if you lower write_throttle, user write performance may suffer, but the number of dirty buffers is limited, so sync operations complete much faster.

Because lowering write_throttle may in some cases delay write requests (for example, lowering write_throttle may increase the file disk queue to the max_diskq value, delaying user writes until the disk queue decreases), it is advisable not to change the value of write_throttle unless your system has a combination of large physical memory and slow storage devices.

Troubleshooting File System Problems

There are three different ways how a file system reacts to problems:

- **marks an inode bad**
  this happens if an inode update or a directory- block update fails. In this case any attempt to access the data in the file or change the inode will fail.

  Inodes can be marked bad if an inode update or a directory-block update fails.
  In these types of failures, the file system doesn't know what information is on the disk, and considers all the information that it finds to be invalid. After an inode is marked bad, the kernel still permits access to the file name, but any attempt to access the data in the file or change the inode fails.
- **disables transactions**
  this happens if the file system detects an error while writing to the intent log. In this case block or inode frees or allocation, structural changes, directory entry changes will fail.

  If the file system detects an error while writing the intent log, it disables transactions. After transactions are disabled, the files in the file system can still be read or written, but no block or inode frees or allocations, structural changes, directory entry changes, or other changes to metadata are allowed.
- **disables the file system**
  this happens if a superblock update fails.

  If an error occurs that compromises the integrity of the file system, VxFS disables itself. If the intent log fails or an inode-list error occurs, the super-block is ordinarily updated (setting the VX_FULLFSCK flag) so that the next fsck does a full structural check. If this super-block update fails, any further changes to the file system can cause inconsistencies that are undetectable by the intent log replay.
  To avoid this situation, the file system disables itself.

Anytime the file system encounters a problem it will write an error message to the syslog.log In nearly every case the file system must be umounted and a fsck has to be performed.
The main reason for file system errors result from hardware failures, hence it is recommended to

check the syslog for I/O errors.

JFS Error Codes

Most messages can appear in one of two forms. The first form applies to the file system in general. The second form is specific to the structural file set within the file system.
Messages of the second form contain the string (structural) to indicate that they occurred in the structural file set.
These errors are categorized in the following areas:

- **File System Response to Problems**
- **Marking an Inode Bad**
- **Disabling Transactions**
- **Disabling the File System**
- **Recovering a Disabled File System**
- **Kernel Messages**
- **Global Message IDs**

Recovering a Disabled File System

When the file system is disabled, no data can be written to the disk. Although some minor file system operation still work, most simply return EIO. The only thing that can be done when the file system is disabled is to do a umount and run a full fsck.
Although a log replay may produce a clean file system, do a full structural check to be safe.
To do a full structural check, use the following example :

 # fsck -F vxfs -o full, nolog -y /dev/rdsk/c1t0d0s1

The file system usually becomes disabled because of disk errors.
Disk failures that disabled a file system should be fixed as quickly as possible (see ***fsck_vxfs***(1M)).

VxFS Kernel Messages

This section lists the VxFS kernel error messages in numerical order.
The Explanation sub-section for each message describes the problem.
The Action sub-section suggests possible solutions.

Global Message IDs

Each time a VxFS kernel message is displayed on the system console, it is displayed along with a monatomically increasing message ID, shown in the *msgcnt* field. This ID guarantees that the sequence of events is known in order to help analyze file system problems.
Each message is also written to an internal kernel buffer and can be viewed in the file

/var/adm/messages.
In some cases, additional data is written to the kernel buffer. For example, if an inode is marked bad, the contents of the bad inode is written. When an error message is displayed on the console, you can use the unique message ID to find the message in /var/adm/messages and obtain the additional information.

Message: 001

NOTICE: msgcnt x: vxfs: mesg 001: vx_nospace - mount_point file system full (n block extent)
Explanation:
The file system is out of space. Often, there is plenty of space and one runaway process used up all the remaining free space. In other cases, the available free space becomes fragmented and unusable for some files.
Action: Monitor the free space in the file system and prevent it from becoming full. If a runaway process has used up all the space, stop that process, find the files created by the process, and remove them. If the file system is out of space, remove files, defragment, or expand the file system.
To remove files, use the find command to locate the files that are to be removed. To get the most space with the least amount of work, remove large files or file trees that are no longer needed.
To defragment or expand the file system, use ***fsadm*** (see the fsadm(1M) manual page).

Message: 002

WARNING: msgcnt x: vxfs: mesg 002:vx_snap_strategy - mount_point file system write attempt to read-only file system
WARNING: msgcnt x: vxfs: mesg 002: vx_snap_copyblk - mount_point file system write attempt to read-only file system

Explanation:
The kernel tried to write to a read-only file system. This is an unlikely problem, but if it occurs, the file system is disabled.
Action: The file system was not written, so no action is required. Report this as a bug to your customer support organization.

Message: 003, 004, 005

WARNING: msgcnt x: vxfs: mesg 003: vx_mapbad - mount_point file system free extent bitmap in au aun marked bad
WARNING: msgcnt x: vxfs: mesg 004: vx_mapbad - mount_point file system free inode bitmap in au aun marked bad
WARNING: msgcnt x: vxfs: mesg 005: vx_mapbad - mount_point file system inode extended operation bitmap in au aun marked bad

Explanation:
If there is an I/O failure while writing a bitmap, the map is marked bad. The kernel considers the maps to be invalid, so does not do any more resource allocation from maps. This situation can cause the file system to report "out of space" or "out of inode" error messages even though df may report an adequate amount of free space. This error may also occur due to bitmap inconsistencies. If a bitmap fails a consistency check, or blocks are freed that are already free in

the bitmap, the file system has been corrupted. This may have occurred because a user or process wrote directly to the device or used *fsdb* to change the file system. The VX_FULLFSCK flag is set. If the map that failed was a free extent bitmap, and the VX_FULLFSCK flag can't be set, then the file system is disabled.
Action: Check the console log for I/O errors. If the problem is a disk failure,replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process was writing to the device, report the problem to your customer support organization.
Unmount the file system and use fsck to run a full structural check.

Message: 006, 007

WARNING: msgcnt x: vxfs: mesg 006: vx_sumupd -mount_point file system summary update in au aun failed
WARNING: msgcnt x: vxfs: mesg 007: vx_sumupd - mount_point file system summary update in inode au iaun failed

Explanation:
An I/O error occurred while writing the allocation unit or inode allocation unit bitmap summary to disk. This sets the VX_FULLFSCK flag on the file system. If the VX_FULLFSCK flag can't be set, the file system is disabled.
Action: Check the console log for I/O errors. If the problem was caused by a disk failure, replace the disk before the file system is mounted for write access, and use fsck to run a full structural check.

Message: 008, 009

WARNING: msgcnt x: vxfs: mesg 008: vx_direrr - mount_point file system inode inumber block blkno error errno
WARNING: msgcnt x: vxfs: mesg 009: vx_direrr - mount_point file system inode inumber immediate directory error errno

Explanation:
A directory operation failed in an unexpected manner. The mount point,inode, and block number identify the failing directory. If the inode is an immediate directory, the directory entries are stored in the inode, so no block number is reported. If the error is ENOENT or ENOTDIR, an inconsistency was detected in the directory block. This inconsistency could be a bad free count, a corrupted hash chain, or any similar directory structure error. If the error is EIO or ENXIO, an I/O failure occurred while reading or writing the disk block. The VX_FULLFSCK flag is set in the super-block so that fsck will do a full structural check the next time it is run.
Action: Check the console log for I/O errors. If the problem was caused by a disk failure, replace the disk before the file system is mounted for write access. Unmount the file system and use fsck to run a full structural check.

Message: 010

WARNING: msgcnt x: vxfs: mesg 010: vx_ialloc - mount_point file system inode

inumber not free

Explanation:
When the kernel allocates an inode from the free inode bitmap, it checks the mode and link count of the inode. If either is non-zero, the free inode bitmap or the inode list is corrupted. The VX_FULLFSCK flag is set in the super-block so that fsck will do a full structural check the next time it is run.
Action: Unmount the file system and use fsck to run a full structural check.

Message: 011

NOTICE: msgcnt x: vxfs: mesg 011: vx_noinode - mount_point file system out of inodes

Explanation: The file system is out of inodes.
Action: Monitor the free inodes in the file system. If the file system is getting full, create more inodes either by removing files or by expanding the file system. .

Message: 012

WARNING: msgcnt x: vxfs: mesg 012: vx_iget - mount_point file system invalid inode number inumber

Explanation:
When the kernel tries to read an inode, it checks the inode number against the valid range. If the inode number is out of range, the data structure that referenced the inode number is incorrect and must be fixed. The VX_FULLFSCK flag is set in the super-block so that fsck will do a full structural check the next time it is run.
Action: Unmount the file system and use fsck to run a full structural check.

Message: 013

WARNING: msgcnt x: vxfs: mesg 013: vx_iposition - mount_point file system inode inumber invalid inode list extent

Explanation: **This is a very serious error.**
For a Version 2 and above disk layout, the inode list is dynamically allocated. When the kernel tries to read an inode, it must look up the location of the inode in the inode list file. If the kernel finds a bad extent, the inode can't be accessed. All of the inode list extents are validated when the file system is mounted, so if the kernel finds a bad extent, the integrity of the inode list is questionable. The VX_FULLFSCK flag is set in the super-block and the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check.

Message: 014

WARNING: msgcnt x: vxfs: mesg 014: vx_iget - inode table overflow

Explanation:
All the system in-memory inodes are busy and an attempt was made to use a new inode.

Action: Look at the processes that are running and determine which processes are using inodes. If it appears there are runaway processes, they might be tying up the inodes. If the system load appears normal, increase the ***vxfs_ninode*** parameter in the kernel .

Message: 015

WARNING: msgcnt x: vxfs: mesg 015: vx_ibadinactive - mount_point file system can't mark inode inumber bad

WARNING: msgcnt x: vxfs: mesg 015: vx_ilisterr - mount_point file system can't mark inode inumber bad

Explanation:

An attempt to mark an inode bad on disk, and the super-block update to set the VX_FULLFSCK flag, failed. This indicates that a catastrophic disk error may have occurred since both an inode list block and the super-block had I/O failures. The file system is disabled to preserve file system integrity.

Action: Unmount the file system and use fsck to run a full structural check.

Check the console log for I/O errors. If the disk failed, replace it before remounting the file system.

Message: 016

WARNING: msgcnt x: vxfs: mesg 016: vx_ilisterr - mount_point file system error reading inode inumber

Explanation:

An I/O error occurred while reading the inode list. The VX_FULLFSCK flag is set.

Action: Check the console log for I/O errors. If the problem was caused by a disk failure, replace the disk before the file system is mounted for write access. Unmount the file system and use fsck to run a full structural check.

Message: 017

WARNING: msgcnt x: vxfs: mesg 017: vx_attr_getblk - mount_point file system inode inumber marked bad

WARNING: msgcnt x: vxfs: mesg 017: vx_attr_iget - mount_point file system inode inumber marked bad

WARNING: msgcnt x: vxfs: mesg 017: vx_attr_indadd - mount_point file system inode inumber marked bad

WARNING: msgcnt x: vxfs: mesg 017: vx_attr_indtrunc - mount_point file system inode inumber marked bad

WARNING: msgcnt x: vxfs: mesg 017: vx_attr_iremove - mount_point file system inode inumber marked bad

WARNING: msgcnt x: vxfs: mesg 017: vx_bmap - mount_point file system inode inumber marked bad

WARNING: msgcnt x: vxfs: mesg 017: vx_bmap_indirect_ext4 - mount_point file

system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_delbuf_flush - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_dio_iovec - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_dirbread - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_dircreate - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_dirlook - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_doextop_iau - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_doextop_now - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_do_getpage - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_enter_ext4 - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_exttrunc - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_get_alloc - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_ilisterr - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_ilock - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_indtrunc - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_iread - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_iremove - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_iremove_attr - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_logwrite_flush - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_oltmount_iget - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_overlay_bmap - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_readnomap - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_reorg_trunc - mount_point file system inode inumber marked bad

WARNING: msgcnt x: vxfs: mesg 017: vx_stablestore - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_tranitimes - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_trunc - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_write_alloc2 - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_write_default - mount_point file system inode inumber marked bad
WARNING: msgcnt x: vxfs: mesg 017: vx_zero_alloc - mount_point file system inode inumber marked bad

Explanation:
When inode information is no longer dependable, the kernel marks it bad on disk. The most common reason for marking an inode bad is a disk I/O failure. If there is an I/O failure in the inode list, on a directory block, or an indirect address extent, the integrity of the data in the inode, or the data the kernel tried to write to the inode list, is questionable. In these cases, the disk driver prints an error message and one or more inodes are marked bad. The kernel also marks an inode bad if it finds a bad extent address, invalid inode fields, or corruption in directory data blocks during a validation check. A validation check failure indicates the file system has been corrupted. This usually occurs because a user or process has written directly to the device or used fsdb to change the file system. The VX_FULLFSCK flag is set in the super-block so fsck will do a full structural check the next time it is run.
Action: Check the console log for I/O errors. If the problem is a disk failure, replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process is writing to the device, report the problem to your customer support organization.
In either case, unmount the file system and use fsck to run a full structural check.

Message: 018

WARNING: msgcnt x: vxfs: mesg 018: vx_idelxwri_done - <mount_point> file system inode <inumber> had a write error at offset <off>

Explanation:
This message prints when the file system encounters an error while flushing non-synchronous extending write data. Since the data for the file couldn't be written, the file probably contains stale data. If the file system was mounted with the blkclear mount option, then the VX_AF_NOGROW flag is set on the file. When this flag is set, further attempts to extend the file fail. The flag is cleared when the file is truncated.
Action: Check the console log for I/O errors. If the problem was caused by a disk failure, then the disk should be fixed as soon as possible. If the disk has failed, it should be repaired before the file system is mounted for write access. Make sure the fsck program does a full structural check at the next administrative period.
The *ff* utility should be used to determine the name of the file that failed. The contents of the file should be checked, and if necessary restored from backup or recreated.

Message: 019

WARNING: msgcnt x: vxfs: mesg 019: vx_log_add - mount_point file system log overflow

Explanation:
Log ID overflow. When the log ID reaches VX_MAXLOGID (approximately one billion by default), a flag is set so the file system resets the log ID at the next opportunity. If the log ID has not been reset, when the log ID reaches VX_DISLOGID (approximately VX_MAXLOGID plus 500 million by default), the file system is disabled. Since a log reset will occur at the next 60 second sync interval, this should never happen.
Action: Unmount the file system and use fsck to run a full structural check.

Message: 020

WARNING: msgcnt x: vxfs: mesg 020: vx_logerr - mount_point file system log error errno

Explanation:
Intent log failed. The kernel will try to set the VX_FULLFSCK and VX_LOGBAD flags in the super-block to prevent running a log replay. If the super-block can't be updated, the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check. Check the console log for I/O errors. If the disk failed, replace it before remounting the file system.

Message: 021

WARNING: msgcnt x: vxfs: mesg 021: vx_mountsetup - mount_point file system validation *failure*

Explanation:
When a VERITAS File System is mounted, the structure is read from disk. If the file system is marked clean, the structure is correct and the first block of the intent log is cleared. If there is any I/O problem or the structure is inconsistent, the kernel sets the VX_FULLFSCK flag and the mount fails. If the error isn't related to an I/O failure, this may have occurred because a user or process has written directly to the device or used fsdb to change the file system.
Action: Check the console log for I/O errors. If the problem is a disk failure, replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process is writing to the device, report the problem to your customer support organization. In either case, unmount the file system and use fsck to run a full structural check.

Message: 022

WARNING: msgcnt x: vxfs: mesg 022: vx_mountroot - root file system remount failed
Explanation: The remount of the root file system failed. The system will not be usable if the root file system can't be remounted for read/write access. When a VERITAS root file system is

first mounted, it is mounted for read-only access. After fsck is run, the file system is remounted for read/write access. The remount fails if fsck completed a resize operation or modified a file that was opened before the fsck was run. It also fails if an I/O error occurred during the remount. Usually, the system halts or reboots automatically.
Action: Reboot the system. The system either remounts the root cleanly or runs a full structural fsck and remounts cleanly. If the remount succeeds, no further action is necessary. Check the console log for I/O errors. If the disk has failed, replace it before the file system is mounted for write access. If the system won't come up and a full structural fsck hasn't been run, reboot the system on a backup root and manually run a full structural fsck. .

Message: 023

WARNING: msgcnt x: vxfs: mesg 023: vx_unmountroot - root file system is busy and can't be unmounted cleanly

Explanation:
There were active files in the file system and they caused the unmount to fail. When the system is halted, the root file system is unmounted. This happens occasionally when a process is hung and it can't be killed before unmounting the root.
Action: fsck will run when the system is rebooted. It should clean up the file system. No other action is necessary.

Message: 024

WARNING: msgcnt x: vxfs: mesg 024: vx_cutwait - mount_point file system current usage table update error

Explanation: Update to the current usage table (CUT) failed. For a Version 2 disk layout, the CUT contains a fileset version number and total number of blocks used by each fileset. The VX_FULLFSCK flag is set in the super-block. If the super-block can't be written, the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check.

Message: 025

WARNING: msgcnt x: vxfs: mesg 025: vx_wsuper - mount_point file system superblock update failed

Explanation:
An I/O error occurred while writing the super-block during a resize operation. The file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check.
Check the console log for I/O errors. If the problem is a disk failure,replace the disk before the file system is mounted for write access.

Message: 026

WARNING: msgcnt x: vxfs: mesg 026: vx_snap_copyblk - mount_point primary file system read error

Explanation:
Snapshot file system error. When the primary file system is written,copies of the original data must be written to the snapshot file system. If a read error occurs on a primary file system during the copy, any snapshot file system that doesn't already have a copy of the data is out of date and must be disabled.

Action: An error message for the primary file system prints. Resolve the error on the primary file system and rerun any backups or other applications that were using the snapshot that failed when the error occurred.

Message: 027

WARNING: msgcnt x: vxfs: mesg 027: vx_snap_bpcopy - mount_point snapshot file system write error

Explanation:
A write to the snapshot file system failed. As the primary file system is updated, copies of the original data are read from the primary file system and written to the snapshot file system. If one of these writes fails, the snapshot file system is disabled.
Action: Check the console log for I/O errors. If the disk has failed, replace it. Resolve the error on the disk and rerun any backups or other applications that were using the snapshot that failed when the error occurred.

Message: 028

WARNING: msgcnt x: vxfs: mesg 028: vx_snap_alloc - mount_point snapshot file system out of space

Explanation:
The snapshot file system ran out of space to store changes. During a snapshot backup, as the primary file system is modified, the original data is copied to the snapshot file system. This error can occur if the snapshot file system is left mounted by mistake, if the snapshot file system was given too little disk space, or the primary file system had an unexpected burst of activity. The snapshot file system is disabled.
Action: Make sure the snapshot file system was given the correct amount of space. If it was, determine the activity level on the primary file system. If the primary file system was unusually busy, rerun the backup. If the primary file system is no busier than normal, move the backup to a time when the primary file system is relatively idle or increase the amount of disk space allocated to the snapshot file system. Rerun any backups that failed when the error occurred.

Message: 029, 030

WARNING: msgcnt x: vxfs: mesg 029: vx_snap_getbp - mount_point snapshot file system block map write error

WARNING: msgcnt x: vxfs: mesg 030: vx_snap_getbp - mount_point snapshot file system block map read error

Explanation:

During a snapshot backup, each snapshot file system maintains a block map on disk. The block map tells the snapshot file system where data from the primary file system is stored in the snapshot file system. If an I/O operation to the block map fails, the snapshot file system is disabled.

Action: Check the console log for I/O errors. If the disk has failed, replace it. Resolve the error on the disk and rerun any backups that failed when the error occurred.

Message: 031

WARNING: msgcnt x: vxfs: mesg 031: vx_disable - mount_point file system disabled

Explanation:

File system disabled, preceded by a message that specifies the reason.

This usually indicates a serious disk problem.

Action: Unmount the file system and use fsck to run a full structural check. If the problem is a disk failure, replace the disk before the file system is mounted for write access.

Message: 032

WARNING: msgcnt x: vxfs: mesg 032: vx_disable - mount_point snapshot file system disabled

Explanation:

Snapshot file system disabled, preceded by a message that specifies the reason.

Action: Unmount the snapshot file system, correct the problem specified by the message, and rerun any backups that failed due to the error.

Message: 033

WARNING: msgcnt x: vxfs: mesg 033: vx_check_badblock - mount_point file system had an I/O error, setting VX_FULLFSCK

Explanation:

When the disk driver encounters an I/O error, it sets a flag in the super-block structure. If the flag is set, the kernel will set the VX_FULLFSCK flag as a precautionary measure. Since no other error has set the VX_FULLFSCK flag, the failure probably occurred on a data block.

Action: Unmount the file system and use fsck to run a full structural check. Check the console log for I/O errors. If the problem is a disk failure, replace the disk before the file system is

mounted for write access.

Message: 034

WARNING: msgcnt x: vxfs: mesg 034: vx_resetlog - mount_point file system can't reset log

Explanation: The kernel encountered an error while resetting the log ID on the file system. This happens only if the super-block update or log write encountered a device failure. The file system is disabled to preserve its integrity.
Action: Unmount the file system and use fsck to run a full structural check. Check the console log for I/O errors. If the problem is a disk failure,replace the disk before the file system is mounted for write access.

Message: 035

WARNING: msgcnt x: vxfs: mesg 035: vx_inactive - mount_point file system inactive of locked *inode inumber*

Explanation:
VOP_INACTIVE was called for an inode while the inode was being used. This should never happen, but if it does, the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check. Report as a bug to WTEC.

Message: 036

WARNING: msgcnt x: vxfs: mesg 036: vx_lctbad - mount_point file system link count table *lctnumber bad*

Explanation:
Update to the link count table (LCT) failed. For a Version 2 and above disk layout, the LCT contains the link count for all the structural inodes. The VX_FULLFSCK flag is set in the super-block. If the super-block can't be written, the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check.

Message: 038

WARNING: msgcnt x: vxfs: mesg 038: vx_dqbad - mount_point file system quota file update error for id

Explanation:
An update to the user quotas file failed for the user ID. The quotas file keeps track of the total number of blocks and inodes used by each user, and also contains soft and hard limits for each user ID. The VX_FULLFSCK flag is set in the super-block. If the super-block cannot be written, the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check. Check the console

log for I/O errors. If the disk has a hardware failure, it should be repaired before the file system is mounted for write access.

Message: 039

WARNING: msgcnt x: vxfs: mesg 039: vx_dqget - mount_point file system user quota file can't read quota for id

Explanation:
A read of the user quotas file failed for the uid. The quotas file keeps track of the total number of blocks and inodes used by each user, and contains soft and hard limits for each user ID.
The VX_FULLFSCK flag is set in the super-block. If the super-block cannot be written, the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check. Check the console log for I/O errors. If the disk has a hardware failure, it should be repaired before the file system is mounted for write access.

Message: 040

WARNING: msgcnt x: vxfs: mesg 040: vx_bsdquotaupdate - mount_point file system user id disk limit reached.

Explanation:
The hard limit on blocks was reached. Further attempts to allocate blocks for files owned by the user will fail.
Action: Remove some files to free up space.

Message: 041

WARNING: msgcnt x: vxfs: mesg 041: vx_bsdquotaupdate - mount_point file system user id disk quota exceeded too long

Explanation:
The soft limit on blocks was exceeded continuously for longer than the soft quota time limit. Further attempts to allocate blocks for files will fail.
Action: Remove some files to free up space.

Message: 042

WARNING: msgcnt x: vxfs: mesg 042: vx_bsdquotaupdate - mount_point file system user id disk quota exceeded.

Explanation: The soft limit on blocks is exceeded. The soft limit can be exceeded for a certain amount of time before allocations begin to fail. Once the soft quota time limit has expired, further attempts to allocate blocks for files will fail.
Action: Remove some files to free up space.

Message: 043

WARNING: msgcnt x: vxfs: mesg 043: vx_bsdiquotaupdate - mount_point file system user id inode limit reached.

Explanation:
The hard limit on inodes was exceeded. Further attempts to create files owned by the user will fail.
Action: Remove some files to free inodes.

Message: 044

WARNING: msgcnt x: vxfs: mesg 044: vx_bsdiquotaupdate - mount_point file system user id inode quota exceeded too long

Explanation:
The soft limit on inodes has been exceeded continuously for longer than the soft quota time limit. Further attempts to create files owned by the user will fail.
Action: Remove some files to free inodes.

Message: 045

WARNING: msgcnt x: vxfs: mesg 045: vx_bsdiquotaupdate - warning: mount_point file system user id inode quota exceeded

Explanation: The soft limit on inodes was exceeded. The soft limit can be exceeded for a certain amount of time before attempts to create new files begin to fail. Once the time limit has expired, further attempts to create files owned by the user will fail.
Action: Remove some files to free inodes.

Message: 046, 047

WARNING: msgcnt x: vxfs: mesg 046: vx_dqread - warning: mount_point file system external user quota file read failed

WARNING: msgcnt x: vxfs: mesg 047: vx_dqwrite - warning: mount_point file system external user quota file write failed.

Explanation:
To maintain reliable usage counts, VxFS maintains the user quotas file as a structural file in the structural fileset. These files are updated as part of the transactions that allocate and free blocks and inodes. For compatibility with the quota administration utilities, VxFS also supports the standard user visible quota files. When quotas are turned off, synced, or new limits are added, VxFS tries to update the external quota files. When quotas are enabled, VxFS tries to read the quota limits from the external quotas file. If these reads or writes fail, the external quotas file is out of date.
Action: Determine the reason for the failure on the external quotas file and correct it. Recreate the quotas file.

Message: 048

WARNING: msgcnt x: vxfs: mesg 048: vx_mapbad - mount_point file system extent allocation unit state bitmap number marked bad

Explanation:
If there is an I/O failure while writing a bitmap, the map is marked bad. The kernel considers the maps to be invalid, so does not do any more resource allocation from maps. This situation can cause the file system to report "out of space" or "out of inode" error messages even though df may report an adequate amount of free space. This error may also occur due to bitmap inconsistencies. If a bitmap fails a consistency check, or blocks are freed that are already free in the bitmap, the file system has been corrupted. This may have occurred because a user or process wrote directly to the device or used fsdb to change the file system. The VX_FULLFSCK flag is set. If the VX_FULLFSCK flag can't be set, the file system is disabled.
Action: Check the console log for I/O errors. If the problem is a disk failure, replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process was writing to the device, report the problem to your customer support organization. Unmount the file system and use fsck to run a full structural check.

Message: 049

WARNING: msgcnt x: vxfs: mesg 049: vx_esum_bad - mount_point file system extent allocation unit summary number marked bad

Explanation:
An I/O error occurred reading or writing an extent allocation unit summary. The VX_FULLFSCK flag is set. If the VX_FULLFSCK flag can't be set, the file system is disabled.
Action: Check the console log for I/O errors. If the problem is a disk failure, replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process was writing to the device, report the problem to your customer support organization. Unmount the file system and use fsck to run a full structural check.

Message: 050

WARNING: msgcnt x: vxfs: mesg 050: vx_isum_bad - mount_point file system inode allocation unit summary number marked bad

Explanation: An I/O error occurred reading or writing an inode allocation unit summary. The VX_FULLFSCK flag is set. If the VX_FULLFSCK flag can't be set, the file system is disabled.
Action: Check the console log for I/O errors. If the problem is a disk failure, replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process was writing to the device, report the problem to your customer support organization. Unmount the file system and use fsck to run a full structural check.

Message: 051

WARNING: msgcnt x: vxfs: mesg 051: vx_snap_getbitbp - mount_point snapshot file system bitmap write error

Explanation:
An I/O error occurred while writing to the snapshot file system bitmap. There is no problem with the snapped file system, but the snapshot file system is disabled.
Action: Check the console log for I/O errors. If the problem is a disk failure, replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process was writing to the device, report the problem to your customer support organization. Restart the snapshot on an error free disk partition. Rerun any backups that failed when the error occurred.

Message: 052

WARNING: msgcnt x: vxfs: mesg 052: vx_snap_getbitbp - mount_point snapshot file system bitmap read error

Explanation:
An I/O error occurred while reading the snapshot file system bitmap. There is no problem with snapped file system, but the snapshot file system is disabled.
Action: Check the console log for I/O errors. If the problem is a disk failure,replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process was writing to the device, report the problem to your customer support organization. Restart the snapshot on an error free disk partition. Rerun any backups that failed when the error occurred.

Message: 053

WARNING: msgcnt x: vxfs: mesg 053: vx_resize - mount_point file system remount failed

Explanation:
During a file system resize, the remount to the new size failed. The VX_FULLFSCK flag is set and the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check. After the check, the file system shows the new size.

Message: 054

NOTICE: msgcnt x: vxfs: mesg 054: vx_attr_creatop - invalid disposition returned by attribute driver

Explanation:
A registered extended attribute intervention routine returned an invalid return code to the VxFS driver during extended attribute inheritance.
Action: Determine which ***vendor*** supplied the registered extended attribute intervention routine

and contact their customer support organization.

Message: 055

WARNING: msgcnt x: vxfs: mesg 055: vx_metaioerr - file system meta data error

Explanation:

A read or a write error occurred while accessing file system metadata. The full fsck flag on the file system was set. The message specifies whether the disk I/O that failed was a read or a write. File system metadata includes inodes, directory blocks, and the file system log. If the error was a write error, it is likely that some data was lost. This message should be accompanied by another file system message describing the particular file system metadata affected, as well as a message from the disk driver containing information about the disk I/O error.

Action: Resolve the condition causing the disk error. If the error was the result of a temporary condition (such as accidentally turning off a disk or a loose cable), correct the condition. Check for loose cables, etc. Unmount the file system and use fsck to run a full structural check (possibly with loss of data). In case of an actual disk error, if it was a read error and the disk driver remaps bad sectors on write, it may be fixed when fsck is run since fsck is likely to rewrite the sector with the read error. In other cases, you replace or reformat the disk drive and restore the file system from backups. Consult the documentation specific to your system for information on how to recover from disk errors. The disk driver should have printed a message that may provide more information.

Message: 056

WARNING: msgcnt x: vxfs: mesg 056: vx_dataioerr - file system file data error

Explanation: A read or a write error occurred while accessing file data. The message specifies whether the disk I/O that failed was a read or a write. File data includes data currently in files and free blocks. If the message is printed because of a read or write error to a file, another message that includes the inode number of the file will print. The message may be printed as the result of a read or write error to a free block, since some operations allocate an extent and immediately perform I/O to it. If the I/O fails, the extent is freed and the operation fails. The message should be accompanied by a message from the disk driver containing information about the disk I/O error.

Action: Resolve the condition causing the disk error. If the error was the result of a temporary condition (such as accidentally turning off a disk or a loose cable), correct the condition. Check for loose cables,etc. If any file data was lost, restore the files from backups.

Determine the file names from the inode number (see the *ncheck*(1M) manual page for more information.) If an actual disk error occurred, make a backup of the file system, replace or reformat the disk drive, and restore the file system from the backup. Consult the documentation specific to your system for information on how to recover from disk errors. The disk driver should have printed a message that may provide more information.

Message: 057

WARNING: msgcnt x: vxfs: mesg 057: vx_fset_markbad - mount_point file system

mount_point fileset (index number) marked bad

Explanation:
An error occurred while reading or writing a fileset structure. VX_FULLFSCK flag is set. If the VX_FULLFSCK flag can't be set, the file system is disabled.
Action: Unmount the file system and use fsck to run a full structural check.

Message: 058

WARNING: msgcnt x: vxfs: mesg 058: vx_ivalidate - mount_point file system inode number version number exceeds fileset's

Explanation: During inode validation, a discrepancy was found between the inode version number and the fileset version number. The inode may be marked bad, or the fileset version number may be changed, depending on the ratio of the mismatched version numbers. VX_FULLFSCK flag is set. If the VX_FULLFSCK flag can't be set, the file system is disabled.
Action: Check the console log for I/O errors. If the problem is a disk failure, replace the disk. If the problem is not related to an I/O failure, find out how the disk became corrupted. If no user or process is writing to the device, report the problem to your customer support organization. In either case, unmount the file system and use fsck to run a full structural check.

Mounting a JFS file system

 # mount <option> <specific option> /dev/vgXX/lvolX <mountpoint>

You can choose between the following <options>:

-F = file system type
-a = mounts all file systems which are mentioned in /etc/fstab
-e = displays all file systems which are mounted
-l = limit actions to local file systems only
-o = to set specific options

a list of comma-separated sub options and/or keyword/attribute pairs. There are the following specific options:

Miscellaneous <specific options>

rw	read-write (**default**)
suid	set-user ID execution allowed (**default**)
ro	read only
nosuid	set-user-ID execution not allowed
quota	disk quotas enabled
remount	change of mount options for a mounted file system

Intent Log <specific options>

log	all metadata changes are logged immediately (**default**)
delaylog	only critical metadata changes are logged immediately
tmplog	metadata changes logging is almost always delayed
nolog	intend log is disabled

Write <specific_options>

direct	direct writes
dsync	datasynchronous writes
closesync	sync on close writes
tmpcache	temporary caching

Further there is the possibility to determine how ordinary/synchronous writes are treated (only for Online JFS):
mincache = direct|dsync|closesync|tmpcache ordinary writes
convosync = direct|dsync|closesync|tmpcache synchronous writes

Other <specific_options>

blkclear	all data extents are cleared before being allocated to a files

Displaying File System Features
There are two commands available to display the file system features ,for example to check the disk layout or if the file system supports large files.
The *mkfs – m* command displays the command line which created the file system.
Examples:
 # mkfs –m /dev/vgXX/lvolY
 # mkfs -m /dev/vg00/lvol5
 # mkfs -F vxfs -o ninode=unlimited,bsize=1024,version=4,inosize=256,logsize=1024,largefiles /dev/vg00/lvol5 20480

The *fstyp –v* command displays the contents of the superblock
Examples :
 # fstyp –v /dev/vgXX/rlvolY
 # fstyp -v /dev/vg00/rlvol5
vxfs
version: 4
f_bsize: 8192
f_bsize: 8192
f_frsize: 1024
f_blocks: 20480
f_bfree: 18625
f_bavail: 17717

f_files: 3920
f_ffree: 4656
f_favail: 4656
f_fsid: 1073741829
f_basetype: vxfs
f_namemax: 254
f_magic: a501fcf5
f_featurebits: 0
f_flag: 16 <= Largefiles
f_fsindex: 5
f_size: 20480

Features that are Treated Differently between JFS and OnlineJFS

Extension of a file system

Base JFS

 # umount <mountpoint>

 # lvextend –L <n> /dev/vgXX/lvolX -L <n> set new size of lvol to n MB
 # extendfs –F vxfs /dev/vgXX/rlvolX

 # mount <mountpoint>

Online JFS

# lvextend –L <n> /dev/vgXX/lvolX -L <n>	set new size of lvol to n MB
# fsadm –F vxfs –b <n> <mountpoint> -b <n>	set new file system size to n **KB**

Reduction of a file system

Base JFS

Before reduction a backup of the data must be done otherwise the data will be lost

 # umount <mountpoint>

 # lvreduce -L <n> /dev/vgXX/lvolX -L <n> set new size of lvol to n MB

 # newfs –F vxfs /dev/vgXX/rlvolX

 # mount /dev/vgXX/lvolX <mountpoint>

Online JFS

Only file systems which are multiple of 32 MB can be reduced online (it depends on the allocated extent distribution). A backup before execution is recommended

 # fsadm –F vxfs –d –D –e –E <mountpoint>

-d perform directory defragmentation

-e perform extent defragmentation

 # fsadm –F vxfs –b <n> <mountpoint> -b <n> set new file system size to n KB

 # lvreduce –L <n> /dev/vgXX/lvolX -L <n> set new size of lvol to n MB

Enabling large file support

Base JFS

 # umount <mountpoint>

 # fsadm -F vxfs -o largefiles /dev/vgXX/rlvolX

 # mount -F vxfs -o largefiles /dev/vgXX/lvolX <mountpoint>

entry in /etc/fstab (example):

/dev/vgXX/lvolX <mountpoint> vxfs delaylog,largefiles 0 2

check the largefile option: # fsadm -F vxfs <mountpoint>

Online JFS

 # fsadm -F vxfs -o largefiles <mountpoint>

entry in /etc/fstab (example):

/dev/vgXX/lvolX <mountpoint> vxfs delaylog,largefiles 0 2

check the largefile option:

 # fsadm -F vxfs <mountpoint>

Features Only Available for OnlineJFS

Defragmentation of a file system

 # fsadm –F vxfs –d –D –e –E <mountpoint>

-d performs a directory defragmentation
-D reports on directory fragmentation
-e performs a file extent defragmentation
-E reports on file extent fragmentation

Typically directory defragmentation is not beneficial in improving performance.

NOTE: While defragmentation can be run at anytime, it is prudent to do it when the file system is as quiet as possible to reduce the completion time and system impact. It is not advisable to run more than 1 defragmentation at a time.

For those systems with Base JFS , defragmentation is only available by using move (mv) or copy (cp) .
Move is preferable as it incurs less system overhead.

Using extent attributes

Extent attributes can be manipulate using setext

#setext –e <extentsize> -r <reservationsize> -f <flag> <file>

possible flags:

align: Specify that all extents must be aligned on extent_size boundaries relative to the start of allocation units

chgsize: Immediately incorporate the reservation into the file and update the file's on-disk inode with size and block count information that is increased to include the reserved space. The space added to the file is not initialized

contig: Specify that the reservation must be allocated contiguously
noextend: Specify that the file may not be extended after the preallocated space is used
noreserve: Specify that the reservation is not a persistent attribute of the file. Instead, the space

is allocated until the final close of the file, when any space not used by the file is freed. The temporary reservation is not visible to the user

trim: Specify that the reservation is reduced to the current file size after the last close by all processes that have the file open

The extend Attributes can either be seen by the following commands

 # getext <filename>

or

 # ls -le

Creation of a snapshot

1) Creation of a snapshot LV and file systems (should be 10-20% of the snapped file system):

   # lvcreate –L 80 –n lvsnap vgXX –L size of lvol –n name of lvol
   # mount –F vxfs –o snapof=/org_vol /dev/lvsnap /aux_dir /org_vol = file system, which has to backed up /aux_dir = auxiliary directory for mounting snapshot

2) Backup from snapshot file system

   # cd /aux_dir
   # tar cvf /dev/rmt/0m or
   # find . | cpio –ocvx >/dev/rmt/0m

3) Umount the snapshot

   # umount /aux_dir