Quick Reference: AIX Journaled File Systems and Veritas File System October 2000

November 30, 2000

Johnny Shieh

Quick Reference: AIX Journaled File Systems and Veritas File System

October 2000

Quick Reference: AIX Journaled File Systems and Veritas File System

October 2000

Quick Reference: AIX Journaled File Systems and Veritas File System

The purpose of a file system is simply to hold data. UNIX file systems have had a fairly common evolution from their base of Berkeley Software Distribution. It was called the Berkeley Fast File System, which eventually became the UNIX industry standard known as the UNIX File System (UFS). IBM introduced its UNIX file system as the Journaled File System (JFS) with the initial release of AIX Version 3.1. It has now introduced a second file system that is to run on AIX systems called Enhanced Journaled File System (JFS2), which is available in AIX Version 5.0 and later versions. Veritas introduced the Veritas File System (VxFS) that is sold to run as a standalone product, or with Veritas Volume Manager (VxVM) or other volume managers. VxFS can optimize I/O and is designed to work with standard volume managers. JFS and JFS2 are designed to run on the AIX Logical Volume Manager (LVM) that is built into the base of the AIX operating system.

The following commands are available on AIX to manage JFS and JFS2:

backup	Performs a full or incremental backup of a file system
chfs	Changes the characteristics of a file system
crfs	Adds a file system
dd	Copies data directly from one device to another for making file system backups
df	Reports the amount of space used and free on a file system
fsck	Checks file systems and repairs inconsistencies
lsfs	Displays the characteristics of a file system
mkfs	Makes a file system of a specified size on a specified logical volume
mount	Makes a file system available for use
restore	Restores files from a backup
rmfs	Removes a file system
umount	Removes a file system from use

This paper will help administrators of VxFS learn and manage JFS and JFS2. Functional differences and commonality among these file systems will be listed, with a brief discussion about the differences. This paper provides only an overview of differences to the system administrator who is familiar with VxFS systems.

For detailed information about the AIX operating system, refer to the following Web address: http://www.ibm.com/servers/aix/library/.

AIX library information is listed under *Technical Publications*.

Functional Differences

Functional differences between JFS, JFS2, and VxFS include the following:

- Journaling
- I-nodes
- Caching
- Data integrity
- Online File System Resizing
- Online Defragmentation

Journaling

This section discusses logs and their location, as well as mandatory logging.

Logs

A key aspect of any file system is how to recover if the system experiences a system crash. The earlier method, requiring a scan of the entire file system, is time-consuming, and can be a waste of time because only a small portion of the file system requires cleanup after a crash. A solution to this problem is a method of journaling or logging the metadata of files. *Metadata* is everything concerning a file except the actual data inside the file. Elements of the file such as its physical location and size are tracked by the metadata. With logging, whenever something changes in the metadata of a file, this new attribute information will be logged into a reserved area of the file system. Only after the write of the metadata to the log is complete, will the file system write the actual data to the disk platter. If and when a system crash occurs, the system recovery code will analyze the metadata log and try to clean up only those files. Typical cleanup on UNIX systems is through the file system check (**fsck**) command.

An additional feature of VxFS is valid for small, synchronous writes of 8 KB or less. For sufficiently small write sizes that are synchronous in nature, the file system will actually write data, instead of metadata, into the log file or device. The result is that the write requester gets a faster acknowledgement from the file system compared to the metadata-logging method described above. The write data is then transferred from the log file or device to its true file system location at a later time.

Location of Logs

There are internal and external logs. An internal log is the metadata log incorporated into the same physical location as the file system itself. An external log need not be tied to the same disk or physical location as the reserved areas of a file system. Administrators may want the external logs on standalone disks to help improve write throughput, because a write to a file system log must take effect before the actual data write to the file system. Performance is improved because the disk heads are not going to move between the log and the rest of the file system. By default, JFS and JFS2 use external logs. However with JFS2, you can specify the use of an internal log by typing the following with the **mount** command:

```
-a log=INLINE
```

When mandatory logging is set, each file system requires a single log to track it. VxFS requires a single log per file system, although JFS and JFS2 both give the user the option of having one log shared across file systems or giving each file system its own log.

Mandatory Logging

VxFS and JFS allow the user to disable the need to log metadata and to use the log through the mount option. At the present time, this bypass is not available for JFS2. Users can use this option for possible performance enhancement, but it is not the default of the mount option. It must specifically be chosen by the user.

I-nodes

I-nodes are the data structures used to represent files and directories in a file system. Obviously, these entities are the data structures visited most often in a file system. Thus, the access method is of concern for performance reasons. I-node are located by looking up a file name in a directory. A major benefit of using JFS2 is that the i-node is searched using a binary tree to accelerate access.

Caching

This section discusses buffering and direct I/O.

Buffering of Metadata

VxFS allows the user to specify that the metadata, usually written to the log on the platter, not be synchronously written to disk. Instead it is held in memory and the immediate acknowledgment back to the write routine is designed to optimize performance. This implementation has one drawback in cases where the data is still in memory when a system crash occurs. JFS and JFS2 do not allow this caching.

	JFS	**JFS2**	**VxFS**
Cache of metadata	No	No	Yes (optional)

Direct I/O

With direct I/O, the application can bypass kernel memory buffers and move data directly between the disk drive and a user buffer. This feature is available for performance optimization.

	JFS	**JFS2**	**VxFS**
Enabled through program interface	Yes	Yes	Yes
Enabled through mount option	No	No	Yes

Direct I/O is enabled using the program interface on all three file systems. However, VxFS also makes this available using the mount option.

Data Integrity

This section discusses disk scrubbing, flush on close, and errors in metadata.

Disk Scrubbing

Users are sometimes concerned that the area on the platter where they place their data may already contain data bits from previous files. The concern is that, somehow, the file system might confuse “trash” bits with valid data bits. JFS and JFS2 guarantees that unless the data written by the application is written to disk, the block for that data is not allocated to the file. That is, if the data is on disk and you can access it, then it is the data you wrote. (This information is only of interest after a crash or power failure.)

	JFS	JFS2	VxFS
Scrub of disk platter before allocation to file	Always enabled	Always enabled	Mount option

Flush on Close

Flush on close can be done in VxFS.

	JFS	JFS2	VxFS
Close of file automatically flushes data to disk platter	Not offered	Not offered	Special enabled mode

Errors in Metadata

Errors in metadata typically occur as a result of a system crash or a disk platter failing. When errors are detected in metadata in JFS and JFS2, the file system cannot be accessed until the **fsck** command has run to clean up corruption. When errors are detected in metadata in VxFS, it attempts to continue with data that is undamaged. Damaged information is marked and is inaccessible until **fsck** cleanup.

Online File System Resizing

One common problem with operating systems is the need to grow and shrink file systems. LVM and VxVM are the only products that allow their respective file systems to grow online, while the file system is mounted and in use.

However, on the subject of shrinking the file system, JFS2 and VxVM differ. At this time, JFS and JFS2 do not allow the shrinkage of file systems. VxFS does a truncation of the file system. Files that are in the area to be removed are moved to an area within the new size before truncating the file system size.

Online Defragmentation

As with many operating systems, the constant creation, deletion, and modification of files leads to “holes” in the file system where there are gaps of free disk space. All three file systems allow the real-time defragmentation of the file system. That is, the file systems can be mounted and accessed while the defragmentation programs are underway.

With external defragmentation, the program moves portions of files together into the same large disk block locations in order to release larger contiguous blocks of free space for future files.

Internal defragmentation takes external defragmentation further by taking the related data blocks that have been moved together and sorting them so that they are logically ordered to make data access more efficient.

	JFS	JFS2	VxFS
Type of defragmentation	External	External	External and internal

Conclusion

This paper has provided an overview of JFS and JFS2 on AIX and the commands that are available to manage the file system. It also shows some major implementation differences between the Journaled File Systems (JFS and JFS2), which are part of AIX and the Veritas file system (VxFS). In general, both JFS and VxFS provide fast recovery, online growing, online defragmentation, and optimum performance with direct I/O.

Special Notices

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.