SAP Sybase Adaptive Server Enterprise

Raw Devices vs File System

TABLE OF CONTENTS

Introduction

In the UNIX world, a file can be considered as the smallest unit in which information is stored. A file system is usually a collection of files or better still a logical way in which a collection of files are stored and organized for easier management and maintenance. A raw device is a character device which allows accessing a storage device directly without going through the operating system's caches and buffers. Applications like a database management system are capable of using a file system or a raw device for storing and accessing data although they have a slight performance edge when they use raw devices directly enabling them to manage how data is cached instead of depending on the operating system as in the case of a file system.

File system or Raw devices for a Database Management System

It is becoming more and more challenging to make a decision on whether to use a file system or a raw partition device when it comes to a database management system. A lot of factors such as operating system file system implementation, application I/O profiles, available operating system resources such as memory and CPU, operating system tuning for file system cache limits, and swapping preferences have to be taken in to consideration before making a decision on which way to go. This document talks about the pros and cons of using a file system vs raw device in a Sybase ASE environment.

Raw partitions are considered to be good candidates when it comes to performance of activities like writes especially in a high concurrency environment for databases although file system devices were good for large read operations with the file system read-ahead capability. A file system in UNIX uses a buffer cache for disk I/O. This means that the writes to disk are stored in the buffer and may not be written to disk immediately. As far as Sybase ASE is concerned, when it completes a transaction and sends the results to a UNIX file, the transaction is considered to be complete even though the UNIX buffer cache may not have been written to disk. This had its own problems as ASE does have any mechanism to confirm whether the writes to disk made it to the disk or not. As a result, if the UNIX system crashes due to some reason, there is a strong likelihood of the Sybase device being corrupt. This was a strong incentive for Sybase to recommend using raw partitions instead of file systems. Placing Sybase devices on raw devices gave Sybase ASE to process its own I/O requests without getting caught in the UNIX buffering scheme and have more control to keep track of what portions of a transactions has successfully completed or failed in the event of a system crash.

How does ASE handle file system and raw partitions

ASE uses file system or raw partitions when a new database device is created. When using a raw partition, it is important to specify the full path to the partition. On the other hand, when using an operating system file, it is ok to use the full path or a relative path. Path names are relative to the server's current working directory. However, Sybase recommends that the full path is specified when referring to path names. Here is an example of creating a database device using **disk init**:

```
disk init name = "user_device1",

physname = "/work/data/device1.dat",

size = 2048
```

In this example, "size = 2048" tells the command to allocate 2048 “virtual” pages to the device. A virtual page is 2048 bytes, so this command creates a 4MB device.

If the existing database device is too small, it is recommended to use the **disk resize** command to make the device larger. This command takes the same “name” and “size” parameters as **disk init**, except the size parameter specifies how much larger the device is desired to be.

disk resize allows dynamically increasing the size of the database devices, rather than initializing a new device. It is possible to use disk resize to increase the size for devices on raw partitions and file systems. The minimum increase on a device is 1MB or an allocation unit, whichever is greater. It is important to keep in mind that the operating system constraints limit how much larger any given device can be made. For example, it is not possible to make a device on a UNIX raw partition larger if the full defined size of that partition has already been allocated.

Sybase ASE introduced DSYNC capabilities for file system support in ASE 12.0.   Although in many of the Sybase documents published prior to ASE 15.5 the recommendation was to use a raw partition, post ASE 15.5, Sybase decided to better support the file system community as well with DIRECT I/O. DIRECT I/O is basically a way to perform I/O on file system devices in a similar way to raw devices i.e. the OS buffer caches are bypassed and data are written directly to disk.

DIRECT I/O does not however guarantee that the writes will only return after all data have been stored on disk (just that data will not go into caches). But since the OS buffer caches are bypassed, it does provide a pretty good recoverability. DIRECT I/O provides better write performance than sync (especially if the device is stored on a SAN). On the other hand, DSYNC is faster on devices for read operations. So transaction log devices are very good candidates for DIRECT I/O (or for raw devices).

The “DSYNC” parameter implemented DSYNC I/O for file system devices, enabling updates to the device to take place directly on the storage media, or buffering them with the UNIX file system. Although it may have appeared that **DSYNC** bypassed the file system buffer to ensure recoverability, it still used the file system buffer, but forced a flush after each file system write. This double buffering in both Adaptive Server Enterprise and the file system cache—plus the flush request—caused slower response times for writes to file system devices than for raw partitions. ASE opens a database device file of a device with the DSYNC setting on, using the operating system DSYNC flag. With this flag, when ASE writes to the device file, both the written data must be physically stored on disk before the system call returns.

This allows for a better recoverability of the written data in case of crash: If the writes are buffered by the OS and the system crashes, these writes are lost. Of course, this only handles OS level buffering. The data could still be in the disk write cache and get lost…

One of the drawbacks of DSYNC is that it costs performance (because the writes, even if buffered by the OS, are guaranteed to go to the disk before the operation finishes). It should be noted that DSYNC doesn't mean that there is not asynchronous I/O. It just means that when you write synchronously or check for whether the asynchronous I/O was performed, you'll only get the response that the write is completed once the data are effectively on the physical disk.

With a 32-bit ASE there was a need for what can be termed as a double-copy or double-buffering. ASE will perform its own caching mechanism and the OS file system will also have its caching mechanism in place which can cause some delay in writing to the disk.   However, with a 64-bit ASE, there is no need for this double-copy or double-buffering concept. Since a 64-bit ASE is capable of handling larger memory, it is prudent to let ASE cache the data and use the allocated memory instead of allowing the OS file system to add another layer of caching.   This provides good performance on a file system like seen on a raw device.

Summary

It is a well-known fact that raw devices provide the highest performance for write activity and ensure data integrity. However, little is said about its inability to read fast – especially sequential reads, and much needed capability of shrinking or growing a device. Although raw devices were considered to be better because of the performance gains, tremendous strides have been made in the file system area that the gap between raw devices and file systems have become negligible. A few examples of the improvements in the file system arena are ease of setting up an application like ASE, ability to take snapshots that can ease lifecycle management, performance benefits and recoverability and more. SAP Business Suite prefers file system to raw partitions especially with the advances in technology and the advantages it brings to the environment.

www.sap.com