developerWorks®

Linux on 4KB-sector disks: Practical advice

Make sure Linux is firing on all cylinders

Roderick W. Smith

April 27, 2010

Starting in December 2009, hard disk manufacturers began introducing disks that use 4096-byte sectors rather than the more common 512-byte sectors. Although this change is masked by firmware that breaks the 4096-byte physical sectors into 512-byte logical sectors for the benefit of the operating system, the use of larger physical sectors has implications for disk layout and system performance. This article examines these implications, including benchmark tests illustrating the likely real-world effects on some common Linux® file systems. As disks with 4096-byte sectors become more common throughout 2010 and beyond, strategies for coping with these new disks will become increasingly important.

Why change to 4096-byte sectors?

If you're familiar with disk structure, you know that disks are broken down into *sectors*, which are normally 512 bytes in size; all read or write operations occur in multiples of the sector size. When you look closer, hard disks actually include a great deal of extra data in between sectors. These extra bytes are used by the disk's firmware to detect and correct errors within each sector. As hard disks grow larger, the result is that more and more data must be stored on each square centimeter of disk, resulting in more low-level errors, thus straining the firmware's error correction capabilities.

One way around this problem is to increase the sector size from 512 bytes to a larger value, enabling more powerful error-correction algorithms to be used. These algorithms can use less data on a per-byte basis to correct for more serious problems than is possible with 512-byte sectors. Thus, changing to a larger sector size has two practical benefits: improved reliability and greater disk capacity—at least in theory.

The real-world benefits to end users aren't likely to be obvious in the same way that an increased monitor size or improved central processing unit (CPU) speed are obvious. However, the reduction in space devoted to parity may result in quicker introduction of larger disks or better disk reliability.

Unfortunately, assumptions about a 512-byte sector size lurk throughout the software chain, in tools like the basic input/output system (BIOS), boot loaders, operating system kernels, file system code, and disk utilities. Although a change to a 4096-byte sector size has been brewing for a decade or so, some tools still aren't ready. Microsoft® Windows® XP is often cited as trouble-prone, but even in the Linux world, some problems are only now being corrected.

To help with the transition, the first disks with 4096-byte sectors translate each *physical* sector into eight *logical* sectors of 512 bytes. To the BIOS, the operating system, and all disk utilities, the disk appears to have 512-byte sectors, even though the underlying physical sector size is 4096 bytes. Western Digital, the first manufacturer to release such disks, uses the term *Advanced Format* to refer to disks with 4096-byte physical sectors and software translation to 512-byte logical sectors. This article uses the same term for both Western Digital's disks and future disks using similar technology from other manufacturers.

Why are there performance effects?

Unfortunately, changing the apparent sector size in firmware can degrade performance. To understand why, you should understand something about file system data structures and how partitions are placed on the hard disk.

Most modern file systems use data structures that are 4096 bytes or larger in size. Thus, most disk I/O operations are in multiples of this amount. Consider what happens when Linux wants to read or write one of these data structures on a new disk with 4096-byte sectors. If the file system data structures happen to align perfectly with the underlying physical partition size, a read or write of a 4096-byte data structure results in a read or write of a single sector. The hard disk's firmware doesn't need to do anything extraordinary; but when the file system data structures do not align perfectly with the underlying physical sectors, a read or write operation must access two physical sectors. For a read operation, this takes little or no extra time because the read/write head on the disk most likely passes over both sectors in succession, and the firmware can simply discard the data it doesn't need. Writes of misaligned data structures, on the other hand, require the disk's firmware to first read two sectors, modify portions of both sectors, and then write two sectors. This operation takes longer than when the 4096 bytes occupy a single sector. Thus, performance is degraded.

How can you tell if your data structures are properly aligned? Most file systems align their data structures to the beginning of the partitions that contain them. Thus, if a partition begins on a 4096-byte (8-sector) boundary, it's properly aligned. Unfortunately, until recently, most Linux partitioning tools did not create partitions aligned in this way. The upcoming section, Aligning partitions, describes how to do the job with common Linux partitioning software.

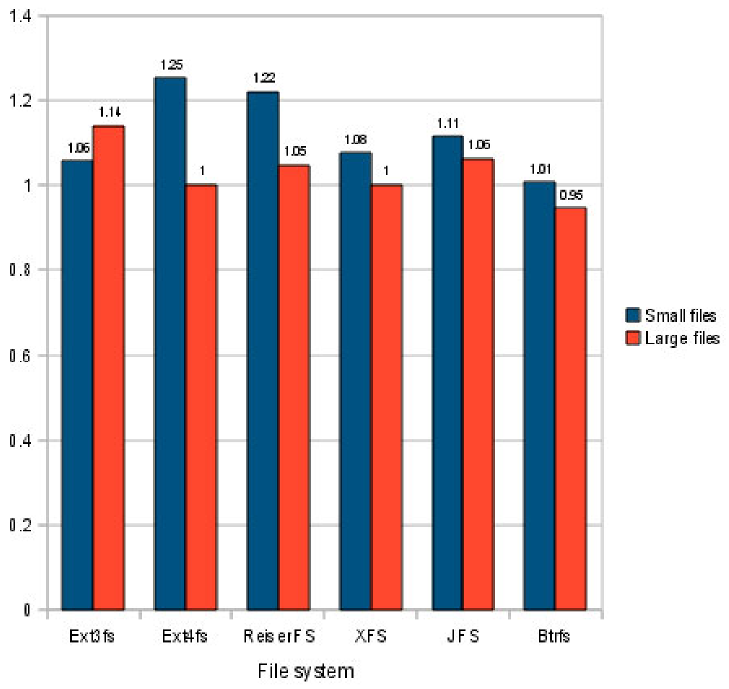
Benchmark results

You might wonder just how important proper partition alignment is. To answer this question, a 1TB Western Digital WD-10EARS Advanced Format drive was used with both aligned and unaligned partitions and several Linux file systems. The disk was partitioned using the globally unique identifier (GUID) Partition Table (GPT) system, with the aligned partitions beginning at logical sector 40 and the unaligned partitions beginning at logical sector 34 (the first available sector when using a GPT disk with its default partition table size). File systems tested were ext3fs, ext4fs, ReiserFS (version 3), JFS, XFS, and Btrfs. The computer ran a 64-bit 2.6.32.3 Linux kernel.

A script performed a series of disk I/O operations, including creating a fresh file system, extracting an uncompressed Linux kernel tarball to the test drive, copying the tarball to the drive, reading the just-uncompressed files on the test drive, reading the tarball from the drive, and removing the Linux kernel directory. The source Linux kernel tarball was stored on another disk, and for read tests, output was directed to /dev/null. After each write test, the test disk was unmounted as a way to ensure that no operations remained in Linux's disk cache. Figures reported include the time required to perform the unmount operation. The kernel tarball was 365MB in size—far larger than the disk's 64MB cache. Each test sequence was run six times for each file system, three times on properly aligned and three times on improperly aligned partitions. Between-run variability was small. The average unaligned time was divided by the average aligned time to determine how much of a performance hit improper alignment imposed. A value above 1.00 indicates some performance penalty for improper alignment.

Many of the tests produced modest impairment. The values for file system creation ranged from 0.96 (for XFS) to 7.94 (for ReiserFS), with a mean value of 2.79. Because file system creation is normally done only rarely, this impairment is not that important. The read tests produced ratios ranging from 0.95 to 1.25, indicating no more than a 25% speed penalty, as detailed in Figure 1. A value of 1.00 means no penalty; higher values mean worse performance.

Figure 1. Read performance penalty for using unaligned partitions

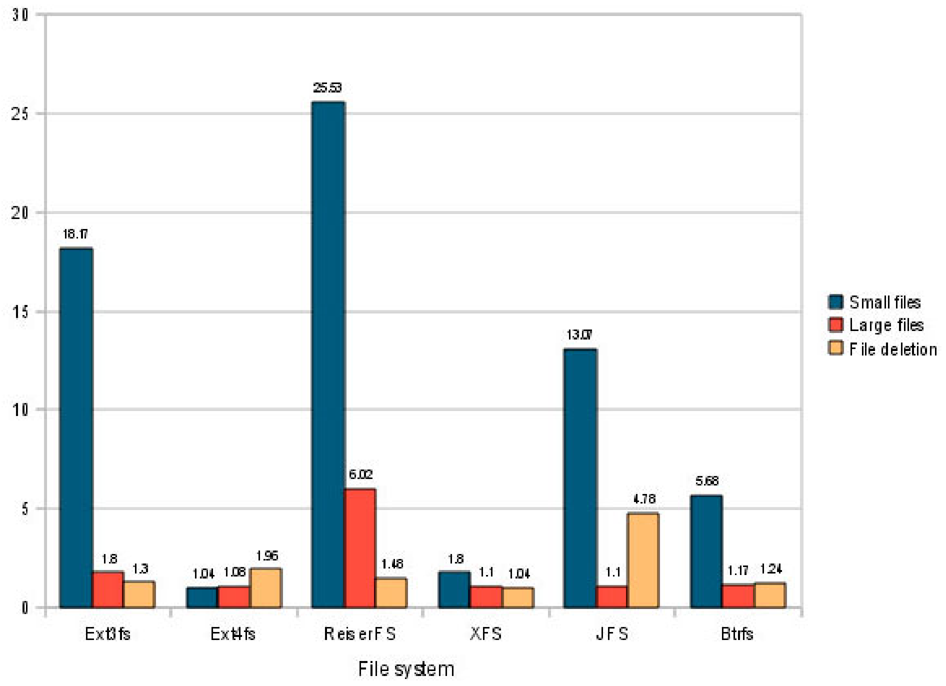


Large-file write performance also suffered only modest impairment. These values ranged from 1.10 (for XFS and JFS) to 6.02 (for ReiserFS), with a mean of 2.10. Much of that elevation is attributable to ReiserFS's sensitivity. Removing it produces a mean of 1.31 for the remaining five file systems. File deletion effects were similar, ranging from 1.04 (for XFS) to 4.78 (for JFS), with a mean of 1.97. Removing JFS as an outlier produces a mean of 1.40.

The biggest write performance effects occurred with small file creation (extracting the kernel tarball). Effects on tarball extraction ranged from 1.04 (for ext4fs) to 25.53 (for ReiserFS), with a mean of 10.9. The second-best performer in this test was XFS, with a figure of 1.82. Because these figures are ratios of unaligned to aligned performance, a value of 10.9 means that a tarball extraction that takes 10 seconds on a properly aligned partition takes 109 seconds on an improperly aligned partition—a huge difference! The XFS figure of 1.82 means that this 10-second operation takes 18.2 seconds on an improperly aligned partition.

Figure 2 summarizes these write performance impairments across all file systems. As before, a value of 1.00 means no penalty; higher values mean worse performance.

Figure 2. Write performance penalty for using unaligned partitions



Note that these tests do not reflect overall performance across file systems. You should not conclude, for instance, that ReiserFS is a poor performer because it produced some of the largest performance differences. ReiserFS is, however, more sensitive than most others to improper alignment.

In addition to running tests on file systems in partitions, a spot check was done on file systems in an LVM configuration, with the LVM partition being either properly or improperly aligned. These results were similar to those of the raw partition results.

What does all this mean, as a practical matter? You should begin by determining the physical sector size of your disk. If you find that you've got an Advanced Format drive, you should align your partitions properly.

Determining physical sector size

In theory, the Linux kernel should return information on the physical sector size in the /sys/block/sdX/queue/physical_block_size pseudo-file and on the logical sector size in the /sys/block/sdX/queue/logical_block_size pseudo-file, where sdX is your device's node name (normally sda, sdb, and so on). In practice, however, the physical block size information is spurious, at least for the first generation of Western Digital Advanced Format drives. Unfortunately, this means that disk utilities cannot properly detect the presence of such disks.

As a practical matter, then, you must look up your drive's specifications on the manufacturer's Web site or in other ways. The /sys/block/sdX/device/model pseudo-file holds the device's model number, so you can look there and then check with the manufacturer.

With the current first generation of Advanced Format drives, Western Digital has placed stickers on the drives advertising the fact that they are Advanced Format drives. Unfortunately, these stickers imply that only Windows XP has problems with these drives. As the above benchmark results reveal, Linux users must exercise extreme caution with these drives.

Aligning partitions

Aligning RAID partitions

Redundant array of independent disks (RAID) levels 5 and 6 have alignment issues similar to those of Advanced Format drives, but the causes relate to the size of data stripes used to create the array, typically 16KB to 256KB. When using RAID arrays, you should align partitions on multiples of the stripe size. The default alignment on 2048 sectors (1024KB) that is emerging as a new standard works well with all common RAID stripe sizes.

Published test results indicate a performance penalty of about 5-30% for improper alignment, which is much less than the penalty for improperly aligning an Advanced Format drive. When creating a RAID array from Advanced Format disks, you don't need to take any extra steps. Because the RAID alignment values are multiples of the 4096-byte alignment required by Advanced Format drives, both technologies' needs are met if you align partitions as for a RAID array of disks with 512-byte physical sectors.

The current Western Digital drives include a jumper you can set for Windows XP compatibility. This jumper has the effect of shifting the sector numbering by 1, thus placing a partition that the computer identifies as beginning on sector 63 (for cylinder alignment) on the true logical sector 64. This is a quick-and-dirty fix to the problem of sector alignment for the common situation in Windows of using a single partition that spans the entire drive. Unfortunately, if you create multiple partitions, anything other than the first will probably be misaligned. Therefore, you should almost certainly *not* use this jumper; instead, use your Linux partitioning software to create properly aligned partitions.

Three families of Master Boot Record (MBR) and GPT partitioning tools are available for Linux, and each offers its own methods of aligning partitions. If you have an Advanced Format drive, your best option is to run the latest Linux partitioning software available.

Tip: If you want to dual-boot between Linux and an older operating system that requires cylinder alignment, try aligning the starts of all your partitions on multiples of eight cylinders. This translates to 8-sector alignment for optimum disk performance as well as cylinder alignment for the older operating system.

`fdisk` family

The `fdisk` family, which ships as part of the `util-linux-ng` package on most distributions, enables fairly direct editing of MBR data structures, but it cannot create or modify file systems. Through `util-linux-ng` 2.17, `fdisk` does not offer any direct support for 8-sector alignment of partitions. Through at least version 2.17.2 (the most recent as I write), the default alignment remains cylinder-based.

You can, however, align partitions properly with any version of `fdisk`. To do this, you should change the default units from cylinders to sectors by typing **`u`** at the main menu. You can then enter start sector values that are multiples of 8. In principle, you can start the first partition with a sector number as low as 8 for proper alignment; however, it's best to start the first partition at 64 or higher to leave room for boot loader code in the unallocated space between the MBR and the first partition. Microsoft's partitioning tools for Windows Vista and Windows 7 begin the first partition at sector 2048, so that's a safe place to start it from a cross-platform point of view. In fact, beginning with `util-linux-ng` 2.17.1, this is the default when you disable DOS compatibility mode by typing **`c`** at the main menu. I recommend doing so.

Note, however, that `fdisk` doesn't align subsequent partitions automatically. Proper alignment is likely to follow if you specify partition size in units of megabytes or larger and then accept the default value for subsequent partitions, but this isn't guaranteed. For safety, you should verify that every partition begins on a multiple of 8.

Another approach with `fdisk` is to launch it as **`fdisk -H 224 -S 56 /dev/sda`**, which changes the cylinder/head/sector (CHS) geometry to guarantee proper 4096-byte alignment when the program aligns to cylinders, as it does by default.

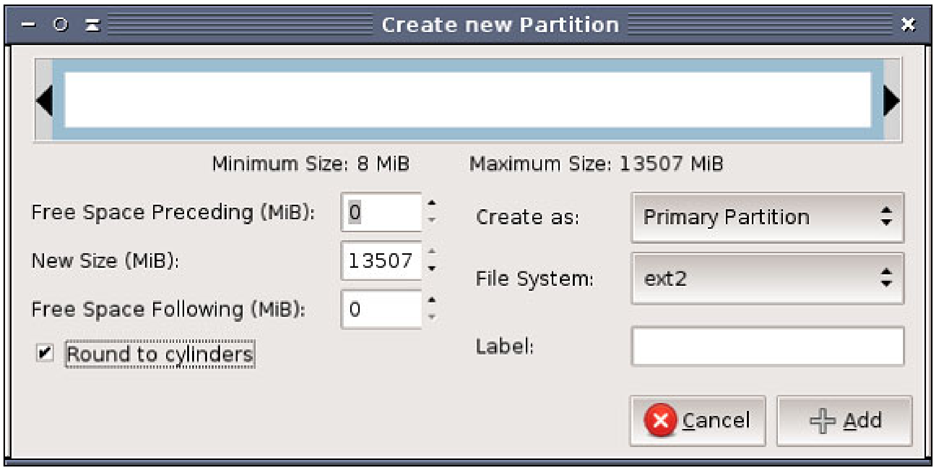
`libparted` library

The `libparted` library powers many Linux partitioning tools that support file system manipulation. Through version 2.1, the text-mode GNU Parted program (command name `parted`) provides little support for aligning on anything but cylinder boundaries. The best approach may be to type **`unit s`** to change the default unit to sectors. You can then manually enter partition start points in sectors and verify partition start points precisely.

Version 2.2 has begun a transition toward a policy that's more useful for disks with 4096-byte physical sectors. With this version, you can specify a starting value of `1M` and the sector will be properly aligned. This version also warns you when your partitions are not properly aligned.

Using the GUI GParted program, you should be sure to clear the "Round to cylinders" check box in the Create New Partition dialog box, shown in Figure 3. You must set the partition's start sector relative to the previous partition's end, but if you begin with a properly aligned partition, this should all work out. You can bring up a partition's Information dialog box to learn what its start and end sectors are in absolute terms.

Figure 3. Clear the "Round to cylinders" check box (here shown selected, which is its default state) when using GParted



Minimum Size: 8 MiB
Maximum Size: 13507 MiB
Free Space Preceding (MiB): 0
New Size (MiB): 13507
Free Space Following (MiB): 0
[✓] Round to cylinders
Create as: Primary Partition
File System: ext2
Label:

GPT fdisk utilities

The GPT fdisk utilities are useful only with GPT disks. Versions prior to 0.5.2 don't perform any alignment, although you can align partitions manually by specifying appropriate start sector numbers. Versions 0.5.2 and 0.6.0 through 0.6.5 adjust the start sectors of all partitions to an 8-sector boundary for large disks (those over about 800GB), but not for smaller disks. Version 0.6.6 introduces a Windows-style 2048-sector (1MB) alignment for all unpartitioned disks and attempts to infer the alignment used in the past on disks with existing partitions.

With versions 0.5.2 and later, you can manually adjust the alignment value via the `l` option on the experts' menu. This option takes a number of sectors as an option. Set it to 8 or a multiple of that amount for proper alignment with Advanced Format disks. The verify option (`v` on any menu) reports any partitions that are not properly aligned based on the current alignment value.

Looking forward

At present, only a handful of Advanced Format hard drive models are available. Press reports indicate that this technology will be spreading to more drives from all the major manufacturers in 2010 and beyond. It's conceivable that new models will suffer other performance problems that differ from those with the first generation of Advanced Format drives.

Ultimately, manufacturers may abandon the fiction of 512-byte sectors, or they may provide jumpers to enable users to choose whether or not to use this compatibility feature. If you encounter a drive with 4096-byte sectors but with an option to use the true sector size, you may want to use it; however, you should be aware of some caveats.

As noted earlier, software from the BIOS up may contain assumptions about a hard disk's sector size. If the BIOS contains such an assumption, it's likely that your computer won't boot from a disk that has 4096-byte sectors and lacks firmware translation to 512-byte sectors. As of version 2.2, GNU Parted displays a warning that support for disks with sectors of other than 512 bytes is experimental when it is launched on such disks. Other problems may lurk in software that may be important to you. Using the latest software may help you work around problems, as may using a conventional disk as the boot disk, restricting your new-technology disk to use as a data disk (/dev/sdb or higher).

Overall, caution is in order when dealing with exotic new disks. That said, chances are that the dust will settle on the current style of Advanced Format disk, as well as other new drive types, relatively quickly.

Related topics

- Develop and deploy your next app on the IBM Bluemix cloud platform.
- "Advanced Format Technology" (Western Digital) is a white paper (PDF), available in several languages, that describes Advanced Format in detail.
- "Exploring WD's Advanced Format HD Technology" (Hot Hardware) includes Windows benchmarks.
- A post by Tejun Heo, Linux kernel developer, describes the technical challenges of Advanced Format drives in Linux software.
- "Linux hardware RAID howto" provides information about Linux RAID alignment issues.
- The GNU Parted Web site hosts both the text-mode GNU Parted and its parent library, libparted. GNU Parted is a mature text-mode MBR and GPT partitioning tool.
- The util-linux-ng package includes Linux `fdisk`, `sfdisk`, and `cfdisk`.
- The GNOME Partition Editor (also known as GParted) is a GUI partitioning tool built atop libparted.
- The GPT fdisk program is a GPT-only partitioning program modeled after Linux `fdisk`.
- In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth other resources for Linux developers and administrators.
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.
- Follow developerWorks on Twitter, or subscribe to a feed of Linux tweets on developerWorks.