Linux on 4 KB sector disks: Practical advice

Make sure Linux is firing on all cylinders

Roderick W. Smith (rodsmith@rodsbooks.com)
Consultant and author

06 March 2014

Advanced Format disks use 4,096-byte sectors rather than the more common 512-byte sectors. This change is masked by firmware that breaks the 4,096-byte physical sectors into 512-byte logical sectors for the benefit of the operating system, but 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 Advanced Format disks have become the norm, understanding how to cope with these disks is a vital skill for anyone who wants to avoid serious performance penalties associated with suboptimal configuration.

Read more by Roderick Smith

Browse all of Roderick's articles on developerWorks. Or search for another author, product, topic, or type of content in our extensive technical library.

If you're familiar with disk structure, you know that disks are broken down into **sectors**, which are typically 512 bytes in size; all Read or Write operations occur in multiples of the sector size. When you look more closely, hard disks include extra data between sectors. The disk uses these extra bytes to detect and correct errors within each sector.

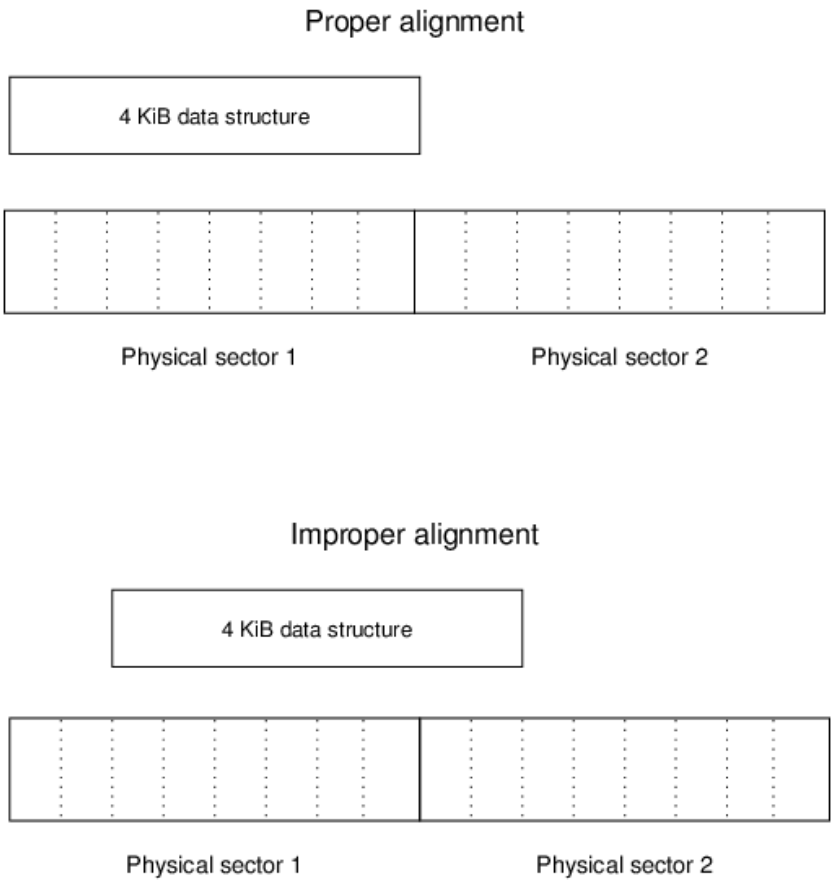
When the sector size is increased from 512 bytes to a larger value, more efficient and powerful error-correction algorithms can be used. Thus, changing to a larger sector size has two practical benefits: improved reliability and greater disk capacity — at least in theory.

Advanced Format disks translate each 4,096-byte **physical** sector into eight 512-byte **logical** sectors. To the firmware, operating system, and all disk utilities, the disk appears to have 512-byte sectors, even though the underlying physical sector size is 4,096 bytes. However, changing the apparent sector size in firmware can degrade performance. To understand why, you need to understand something about file system data structures and how partitions are placed on the hard disk.

Why performance is affected by file system data structures

Most modern file systems use data structures that are 4,096 bytes or larger. 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 4,096-byte sectors. If the file system data structures happen to align perfectly with the underlying physical sector size, a Read or Write of a 4,096-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 don't align perfectly with the underlying physical sectors, a Read or Write operation must access two physical sectors. Figure 1 illustrates this difference:

Figure 1. Disk data structures can straddle physical sectors on an Advanced Format disk



In theory, Read operations should be affected by misalignment less than Write operations. In the case of a disk read, the disk's read/write head is likely to pass over both sectors in quick succession, so the firmware's task of returning the 4 kibibyte (KiB) data structure is relatively simple. In contrast, writes of misaligned data structures require the disk's firmware to read two sectors, modify portions of both sectors, and then write two sectors. This operation takes longer than when the 4,096 bytes occupy a single sector. Thus, performance is degraded. In practice, Read operations can sometimes be affected as badly as Write operations.

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

4,096-byte (eight-sector) boundary, it's properly aligned. However, in the years before 2010 (approximately), Linux partitioning tools didn't create partitions aligned in this way. Even today, some pitfalls remain, so exercise caution when you create your partitions. (To see how to do the job with common Linux partitioning software, go to the upcoming section, Aligning partitions.)

Testing parameters

To learn just how important proper alignment is, I performed tests using three Advanced Format disks on three computers:

- A 1TB Western Digital WD-10EARS Advanced Format drive — one of the first Advanced Format disks introduced (late 2009) on a computer using an NVIDIA MCP61P chip set and a 64-bit 2.6.32.3 kernel. The results from this test appeared in the first version of this article, published in 2010.
- A 2TB Seagate ST2000L003 drive, purchased in 2012, in a computer with a more recent AMD 760G/SB710 chipset and a 64-bit 3.4.1 kernel.
- A 3TB Toshiba DT01ACA300 drive, purchased in late 2013, in a computer with an Intel® H77 chipset and a 64-bit 3.11.7 kernel.

In all three tests, I partitioned the disk 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 the third extended file system (ext3), the fourth extended file system (ext4), ReiserFS (version 3), the Journaled File System (JFS), the Extents File System (XFS), and the B-tree file system (Btrfs).

In all tests, 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 of ensuring that no operations remained in the Linux disk cache. Figures reported include the time required to perform the Unmount operation.

The kernel tarball was 365 mebibytes (MiB) in size for the first test and 451MiB for the second and third tests. All disks had 64MiB caches, so the tarball greatly exceeded the disk's cache size in both tests. I ran each test sequence 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.

Benchmark results

All disks showed impairment for misaligned partitions, with the 2009 Western Digital disk and the 2013 Toshiba disk showing a similar pattern and the 2012 Seagate model displaying a different pattern. Therefore, I describe these results in two groups for each pattern of results.

Western Digital and Toshiba test results

Many of the tests that were run in 2010 on the Western Digital disk 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. On the 2013 Toshiba tests, file system creation values ranged from 1.22 (for ext4) to 1.82 (for both ext3 and XFS), with a mean of 1.57. Because file system creation typically is done only rarely, this impairment is not important. The Western Digital read tests produced ratios ranging from 0.95 to 1.25, indicating no more than a 25% speed penalty, as detailed in Figure 2. A value of 1.00 means no penalty; higher values mean worse performance. The Toshiba values range from 0.94 to 1.11, as shown in Figure 3.

Figure 2. Read performance penalty for using unaligned partitions on a Western Digital WD-10EARS disk

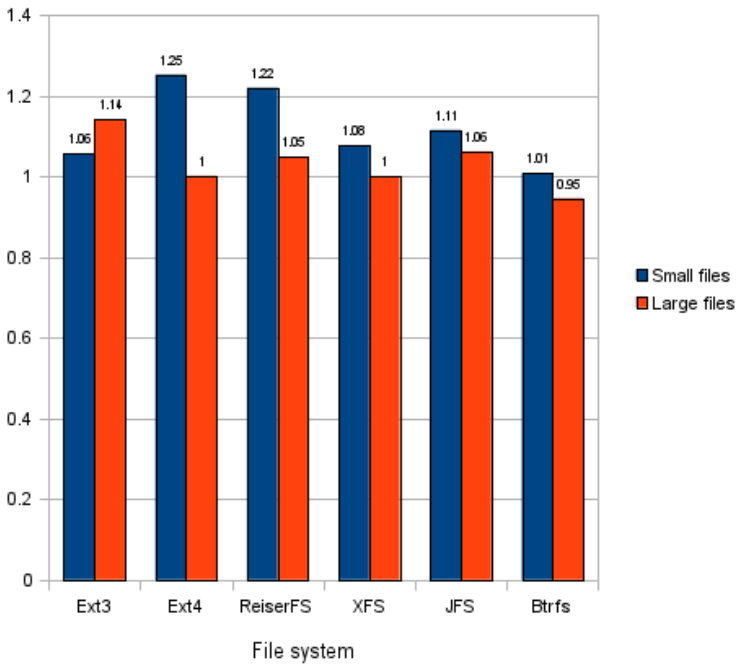
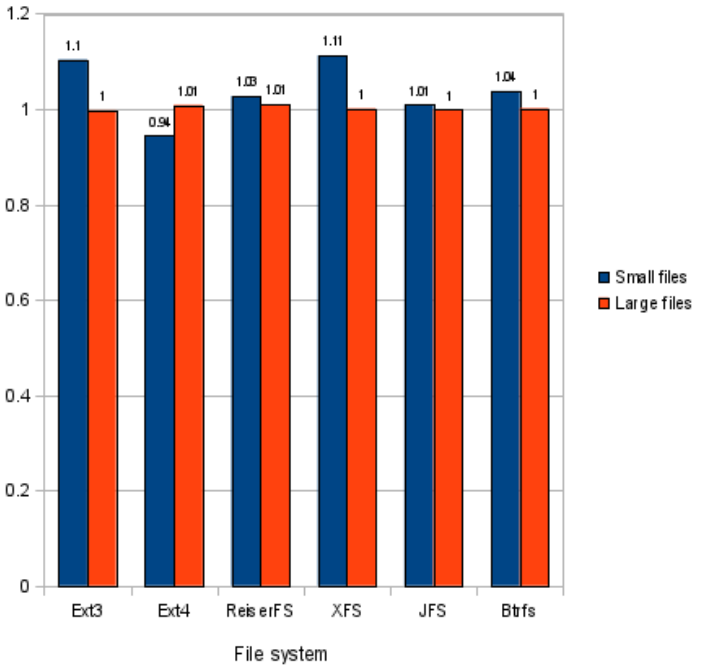


Figure 3. Read performance penalty for using unaligned partitions on a Toshiba DT01ACA300 disk



Large-file write performance also suffered only modest impairment. On the Western Digital disk, these values ranged from 1.10 (for XFS and JFS) to 6.02 (for ReiserFS), with a mean of 2.10; on the Toshiba, they ranged from 1.03 (for ext4) to 2.38 (for ReiserFS), with a mean of 1.34. Much of that elevation is attributable to ReiserFS's sensitivity. Removing it produced means of 1.31 and 1.13 for the remaining five file systems on the Western Digital and Toshiba drives, respectively. File deletion effects were similar. On the Western Digital drive, these ranged from 1.04 (for XFS) to 4.78 (for JFS), with a mean of 1.97; on the Toshiba, the range was from 1.05 (for ext4) to 1.59 (for JFS), with a mean of 1.30.

The biggest write performance effects occurred with small file creation (extracting the kernel tarball). On the Western Digital, effects on tarball extraction ranged from 1.04 (for ext4) to 25.53 (for ReiserFS), with a mean of 10.9. The second-best performer in this test was XFS, with a value of 1.82. On the Toshiba, the effects ranged from 1.44 (for Btrfs) to 3.17 (for ReiserFS), with a mean of 1.92. 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!

Figure 4 summarizes these write performance impairments across all file systems for the Western Digital disk, and Figure 5 does the same for the Toshiba disk. As before, a value of 1.00 means no penalty; higher values mean worse performance.

Figure 4. Write performance penalty for using unaligned partitions on a Western Digital WD-10EARS disk

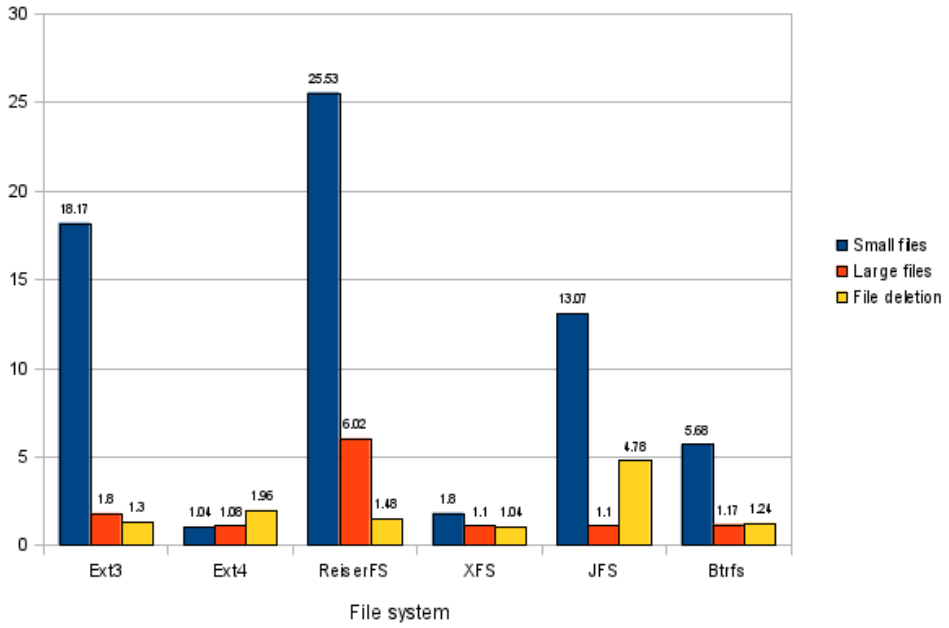
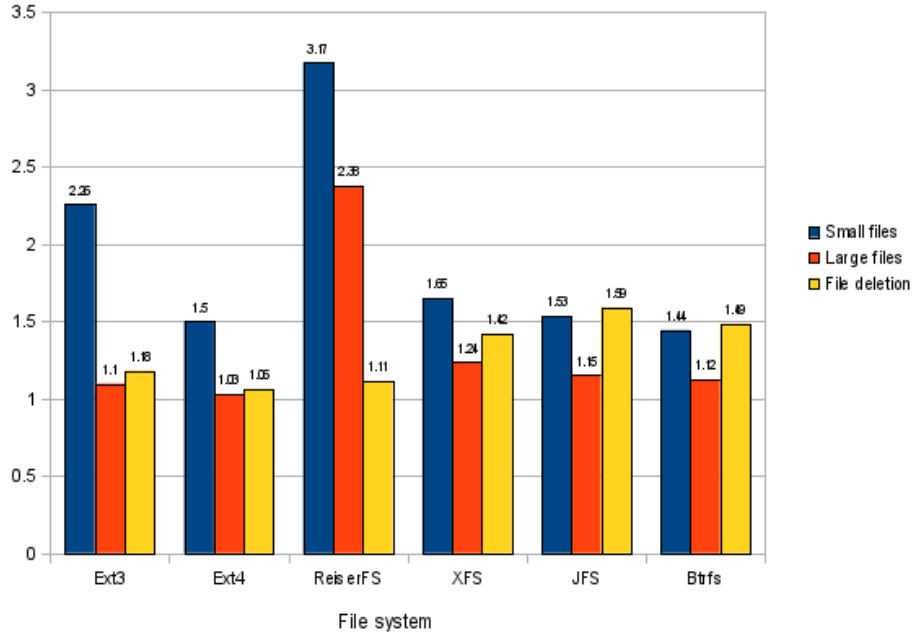


Figure 5. Write performance penalty for using unaligned partitions on a Toshiba DT01ACA300 disk



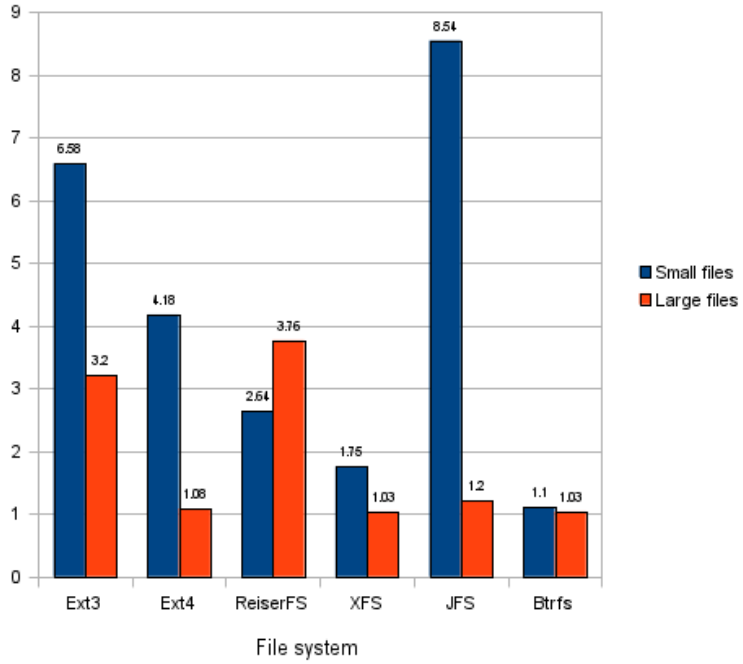
Seagate test results

Tests on the Seagate ST2000L003 disk produced surprisingly different results. The file system creation penalties ranged from 1.09 (for ReiserFS) to 1.97 (for JFS), with a mean of 1.42, which is similar to the Toshiba results.

The surprises begin with the read access results, shown in Figure 6. Read performance suffered significantly more on the Seagate than on the Western Digital or Toshiba drives, ranging as high as

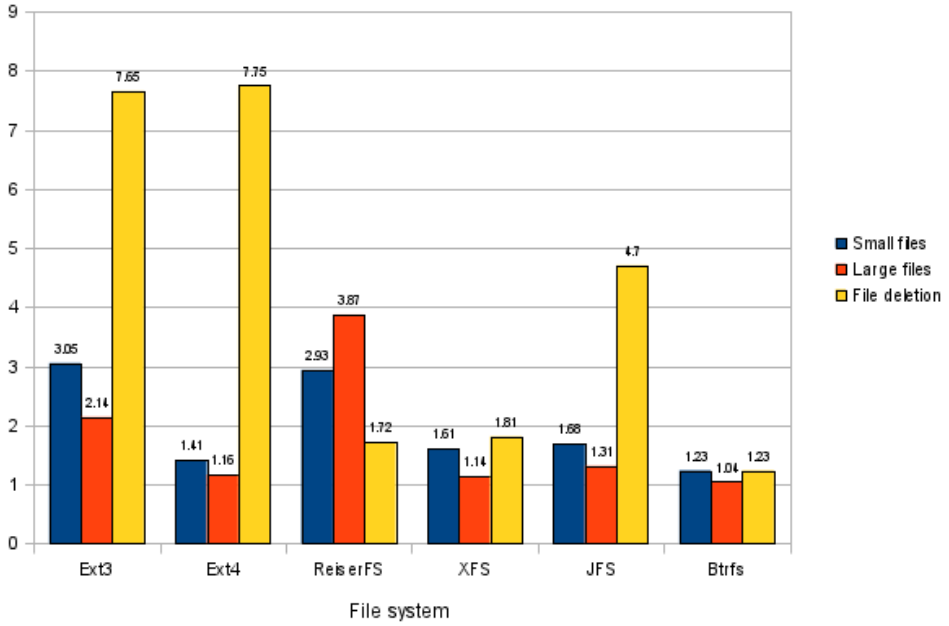
8.54 for small-file reads under JFS, with a mean value of 4.13. Even large-file read performance suffered, with a mean value of 1.88 and a high of 3.76 (for ReiserFS).

Figure 6. Read performance penalty for using unaligned partitions on a Seagate ST2000L003 disk



The Seagate disk's write penalties are shown in Figure 7. Small-file creation penalties ranged from 1.23 (for Btrfs) to 3.04 (for ext3), with a mean of 1.98. Large-file creation penalties ranged from 1.04 (for Btrfs) to 3.87 (for ReiserFS), with a mean of 1.78. For most file systems, the greatest penalty was in file deletions, which ranged from 1.23 (for Btrfs) to 7.75 (for ext4), with a mean of 4.14.

Figure 7. Write performance penalty for using unaligned partitions on a Seagate ST2000L003 disk



Analysis of test results

The difference in the pattern of results among the three hard disks is surprising. Because I conducted these tests over almost four years, the variables (disk brand and model, nondisk hardware, and Linux kernel version) make it impossible to say precisely what caused these differences. I caution against concluding anything about specific disk brands; in particular, the lesser impairment of the Toshiba compared to the Western Digital disk may be a result of improved kernel features or motherboard disk hardware. Ultimately, though, the different pattern of results is unimportant for most people, because the conclusion is the same: Using unaligned partitions produces significant performance impairment.

Note: 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, at least in certain important tests.

If you use a logical volume manager (LVM), be aware that the alignment rules for LVMs are the same as for partitions. Although you needn't be concerned about the alignment of logical volumes within LVMs, you should attend to the alignment of the LVM partitions themselves. A spot check of a few file systems using LVMs replicated the preceding results.

As a practical matter, what does all this mean? 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 (typically, sda, sdb, and so on). In practice, however, the physical block size information is often spurious — in my experience, the kernel reports accurate information only for some disks, and the accuracy of the report also varies with the kernel version. This means that disk utilities can't reliably detect the presence of such disks.

So you may need to look up your drive's specifications on the manufacturer's website 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. If in doubt, assume that your disk is an Advanced Format model — most new disks are.

Western Digital and Toshiba both identify their Advanced Format drives as such with stickers on the drives themselves. However, the Western Digital stickers imply that only Windows XP has problems with these drives, and Toshiba's Advanced Format identification doesn't note any possible performance issues. As my benchmark results reveal, Linux users must exercise extreme caution with these drives. The Seagate Advanced Format drives are not clearly identified as such on their labels.

Aligning partitions

RAID and SSD alignment issues

Redundant array of independent disks (RAID) levels 5 and 6 — and solid-state drives (SSDs) — have alignment issues similar to those of Advanced Format drives. For RAID, alignment should be done to match the size of data stripes used to create the array — typically, 16KiB to 256KiB. For SSDs, alignment should match the erase block size for the drive, which is typically on the order of 512KiB, although I've heard of units with alignment requirements as high as 3MiB. The default alignment of 2,048 sectors (1,024KiB) that's emerging as a new standard works well with most RAID stripe sizes and SSD devices. However, because some SSDs exceed this value, you should check your manufacturer's specifications.

Published test results indicate a performance penalty of about 5% to 30% for improper alignment on RAID arrays, 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 4,096-byte alignment that Advanced Format drives require, the needs of both technologies are met if you align partitions as for a RAID array of disks with 512-byte physical sectors.

Most or all of the Western Digital Advanced Format drives include a jumper you can set for Windows XP compatibility. This jumper shifts the sector numbering by 1, a quick and dirty fix for the common situation in Windows XP of using a single cylinder-aligned partition that spans the entire drive. However, this jumper creates problems if you use multiple partitions or if you use modern partitioning software, so I strongly recommend against trying it. Instead, use your Linux partitioning software to create properly aligned partitions. (Neither Seagate nor Toshiba provides such a jumper on its drives.)

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

The fdisk family

The fdisk family, which ships as part of the util-linux or util-linux-ng package on most distributions, enables fairly direct editing of MBR data structures, but it can't create or modify file systems. Through util-linux version 2.17, fdisk didn't offer any direct support for eight-sector alignment of partitions; alignment remained cylinder-based. This changed with version 2.18, when fdisk began setting the first partition's start point at sector 2,048 by default. If you create all your partitions in fdisk using partition sizes that are multiples of 1MiB or larger, fdisk maintains your partition alignment on 1MiB multiples, which are in turn multiples of eight sectors.

The risk with recent versions of fdisk is that if the disk began with improper alignment, fdisk won't automatically correct for that when you create subsequent partitions. You can also enter manual partition start sectors that are improperly aligned. Thus, when you use fdisk, always check your partition start points to ensure that they're multiples of 8. While you're at it, verify that the program uses sectors as its unit values; even versions later than 2.17 can use cylinders if you type `u` at the main menu. Typing `p` produces a display that you can use to check these details, as shown in Listing 1. Although the unit value is not explicitly stated, it's clear that the units are in sectors, because the start and end values are so large. In this case, the final end value is on the final sector of the disk: Compare it to the total sectors value near the start of the output. Note that fdisk 2.17 and earlier are likely to complain that partitions don't end on cylinder boundaries when they're properly aligned. You can ignore this warning.

An example fdisk output demonstrating proper alignment

```
Command (m for help): p

Disk /dev/sdb: 2000.4 GB, 2000398934016 bytes
256 heads, 63 sectors/track, 242251 cylinders, total 3907029168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Device Boot         Start         End      Blocks   Id  System
/dev/sdb1            2048     2097151     1047552   83  Linux
/dev/sdb2         2097152  3907029167  1952466008   83  Linux
```

When manipulating MBR disks, be aware that the alignment of extended partitions is unimportant. These partitions hold one-sector data structures that define logical partitions, so in a real sense, extended partitions *can't* be properly aligned. Take care to align primary and logical partitions, though.

The libparted library

The libparted library powers many Linux partitioning tools and supports both MBR and GPT partitioning schemes. libparted 3.1 comes with the text-mode parted partitioning tool, and since parted 2.2, you can align to MiB boundaries by specifying start and end points in units of 1MiB

or larger. If you want to verify the alignment, type `unit s` to switch to sector units and check the partition start points, much as you would with fdisk, as shown in Listing 2:

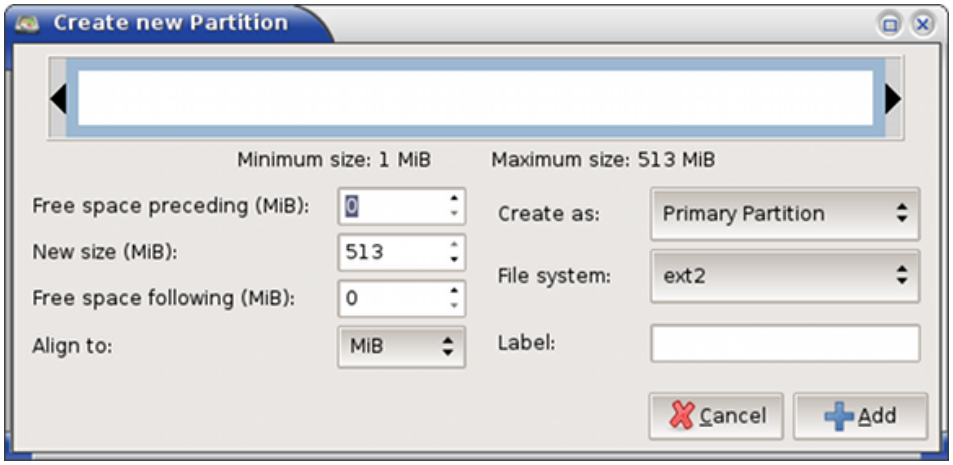
An example parted output demonstrating proper alignment

```
(parted) unit s
(parted) print
Model: ATA ST2000DL003-9VT1 (scsi)
Disk /dev/sdb: 3907029168s
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start       End          Size         File system  Name     Flags
 1      2048s       2097151s     2095104s     ntfs         Windows
 2      2097152s    3907029167s  3904932016s               Linux
```

Using the graphical user interface (GUI) GParted program, be sure to set the **Align to** value to MiB in the **Create new Partition** dialog box, shown in Figure 8. Doing so should produce properly aligned partitions. You can bring up a partition's **Information** dialog box to learn what its start and end sectors are in absolute terms.

Figure 8. Take care to set the Align to value to MiB when creating partitions with GParted



GPT fdisk utilities

The GPT fdisk utilities are useful only with GPT disks. Versions before 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 eight-sector boundary for large disks (those larger than about 800GiB), but not for smaller disks. Version 0.6.6 introduces a Windows-style 2,048-sector (1MiB) 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 with 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 aren't properly aligned based on the current alignment value; `gdisk` displays partition start and end points in sector values. Listing 3 demonstrates use of this program to verify proper partition alignment:

An example gdisk output demonstrating proper alignment

```
Command (? for help): p
Disk /dev/sdb: 3907029168 sectors, 1.8 TiB
Logical sector size: 512 bytes
Disk identifier (GUID): 4B18D328-5E8E-49DB-8690-9FE89807ABF8
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 3907029134
Partitions will be aligned on 8-sector boundaries
Total free space is 6 sectors (3.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1              40          409639   200.0 MiB   8300  Unused /boot
   2          409640          819239   200.0 MiB   8300  Unused /boot
   3          819240      3907029134   1.8 TiB     8E00  Linux LVM

Command (? for help): v

No problems found. 6 free sectors (3.0 KiB) available in 1
segments, the largest of which is 6 (3.0 KiB) in size.
```

Conclusion

At present, the safest assumption is that any new hard disk you buy uses Advanced Format technology. Of course, you can check manufacturer spec sheets to confirm this assumption, but aligning partitions as for an Advanced Format disk has no detrimental effects on older disk types except when using obsolete utilities or operating systems.

Today, some external disks use 4,096-byte sectors, but internal disks use sector size translation. This may change in the future. If you encounter a drive with 4,096-byte sectors but with an option to use the true sector size, you may want to use it; however, be aware of some caveats.

Software from the BIOS up may make assumptions about a hard disk's sector size. If the BIOS contains such an assumption, your computer probably won't boot from a disk that has 4,096-byte sectors and lacks firmware translation to 512-byte sectors. 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.

Resources

Learn

- "Creating a RAID disk array on PowerLinux " (Breno Leitao, developerWorks, January 2013): Read this tutorial to see how you can create a RAID device on PowerLinux machines using an array of disks, including how to identify and format the disks, combine them in a RAID array, create a partition, and create a file system on this partition.
- "Learn Linux, 101: Create partitions and filesystems" (Ian Shields, developerWorks, December 2012): Learn how to create partitions on a disk drive and how to format them for use on a Linux system as swap or data space.
- "Exploring Western Digital's Advanced Format HD Technology" (Joel Hruska, Hot Hardware, February 2010): Read more about this technology, including Windows benchmarks.
- The white paper Advanced Format Technology (Western Digital), available in several languages, describes Advanced Format in detail.
- A post by Tejun Heo, Linux kernel developer, describes the technical challenges of Advanced Format drives in Linux software.
- 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.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners to advanced functionality for experienced developers.
- Follow developerWorks on Twitter.

Get products and technologies

- The GNU Parted website 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 package includes Linux fdisk, sfdisk, and cfdisk.
- The GNOME Partition Editor (also called GParted) is a GUI partitioning tool built on libparted.
- The GPT fdisk program is a GPT-only partitioning program modeled after Linux fdisk.
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, or use a product in a cloud environment.

Discuss

- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Roderick W. Smith

Roderick W. Smith is a consultant and author of more than 20 books on UNIX and Linux, including The Definitive Guide to Samba 3, Linux in a Windows World, and Linux Professional Institute Certification Study Guide. He is also the author of the GPT fdisk partitioning software, and he forked the abandoned rEFIt boot manager to create rEFInd. He resides in Woonsocket, Rhode Island.