

Re: disk write barriers

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/questions/2005-07/0565.html>

From: Matthias Buelow (mkb_at_incubus.de)

Date: 07/07/05

To: freebsd-questions@freebsd.org

Date: Thu, 07 Jul 2005 17:46:04 +0200

Lowell Gilbert <freebsd-questions-local@be-well.ilkk.org> writes:

*>Do you have a background in OS design? It affects the answer, because
>you seem to be referring to access barriers and disk cache flushes
>interchangeably, which doesn't make sense, especially on
>multiprocessor systems.*

>From what I understand from some googling, disk write barriers are specially crafted i/o requests (within the vfs/driver infrastructure) that are acted on by block drivers as following: all requests before a barrier request will be completed before any request that follows the barrier is executed. The driver accomplishes that by issuing the respective flush commands (or uses queue ordering or whatever is supported by the drive) or (in the case of IDE/SATA), may disable and reenable the cache on a barrier. Thus, the barrier maintains an on-disk ordering in requests between "earlier" and "later" requests (otherwise the driver and/or the disk could reorder writes at will).

Since the system doesn't actually run with the cache disabled because it's only used for flushing at sequence points, neither performance nor drive wearout is negatively influenced (noticeably, that is).

An actual application of this is, with journaled filesystems, that the journal will get written to disk before the data is updated. This will guarantee filesystem integrity. From what I understand, MS Windows is doing it that way, and Linux is also using that mechanism (with support for SATA disks only in the latest 2.6 kernel, though.)

*>The problem with caching has nothing to do with flushing the cache; if
>you flush the cache often, there's no advantage to using it anyway.
>The whole speedup from using on-disk caching comes from the fact that
>the drive reorders the writes at will, and lies to the operating
>system by saying the writes are done when they aren't. Among other*

Apparently, performance (and wear&tear) is not overly negatively influenced, since it's used only for periodic flushing after the journal has been written (or potentially, at other events, such as a sync() or

fsync() etc.)

*>obvious problems, this negates the careful ordering calculated by
>softupdates.*

That's where my headaches start.

Softupdates doesn't write a journal at intervals but seems to order writes in general, in a continuous way. Therefore, it would appear that there are no such sequence points. I'm not really aware of the details of how softupdates works, so I'm probably wrong.

I only know that running with the cache disabled seems to be the only safe way to assure that the ordering done by softupdates isn't broken. But disabling the cache is a no-no on modern drives, since they are constructed to be used with the cache on, and disabling it will yield terrible performance and significantly reduce the MTBF, at least on IDE/SATA drives (I'm not talking about enterprise-grade 15krpm SCSI drives, where the situation might be different).

So what's the recommended procedure? Relying on an UPS or that the power will not fail? I mean, I could run fully asynch then and the whole softupdates is of little use, except for (relatively rare) occasions of a kernel crash.

*>This doesn't follow. Just because you know that your drive supports
>disabling the cache does not mean that it is safe to do so.*

Why is that so?

mkb.

freebsd-questions@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-questions>

To unsubscribe, send any mail to "freebsd-questions-unsubscribe@freebsd.org"