

Re: about TRIM/DISCARD support and barriers

Re: about TRIM/DISCARD support and barriers

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-11/msg09468.html>

- *From:* James Bottomley <James.Bottomley@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 24 Nov 2008 14:08:23 -0500
-

On Mon, 2008-11-24 at 19:57 +0100, Jens Axboe wrote:

On Mon, Nov 24 2008, David Woodhouse wrote:

On Mon, 2008-11-24 at 13:42 -0500, James Bottomley wrote:

On Mon, 2008-11-24 at 09:03 +0000, David Woodhouse wrote:

On Mon, 2008-11-24 at 07:52 +0900, James Bottomley wrote:

On Sun, 2008-11-23 at 13:39 +0000, David Woodhouse wrote:

We don't attempt to put non-contiguous ranges into a single TRIM yet.

We don't even merge contiguous ranges -- I still need to fix the elevators to stop writes crossing

Re: about TRIM/DISCARD support and barriers

writes,

I don't think we want to do that ... it's legal if the write isn't a barrier and it will inhibit merging. That may be just fine for a SSD, but it's not for spinning media since they get better performance out of merged writes.

No, I just mean writes `_to the same sector_`.
At the moment, we happily let those cross each other in the queue.

...

It's not a bug ... but changing it might be feasible ... as long as it doesn't affect write performance too much (which I don't think it will), since it is in the critical path.

We could argue about how much sense it makes to let two writes to the same sector actually happen in reverse order.

Especially given the fact that we actually `_do_` preserve ordering in some cases; just not in others. (We preserve ordering only if the start and end of the duplicate writes are `_precisely_` matching; if it's just overlapping (which may well happen in the presence of merges), then this check doesn't trigger.

But that's just semantics. Yes, changing it should be feasible. I talked to Jens about that at the kernel summit, and we agreed that it should probably be done.

The way this currently works is that we sort based on the first sector in the request. So if you have an overlap condition between `rq1` and `rq2` and then a write gets merged into `rq1`, then you may have passing writes. Linux has never guaranteed any write ordering for non-barriers, so we've never attempted to handle it. Direct aliases (matching first sectors) are handled as you mention, but that's more of an algorithmic thing than by design.

My main worry is that this will add considerable overhead to request

Re: about TRIM/DISCARD support and barriers

sorting. For the rbtree based sorting, we'd have to do a rb_next/rb_prev to look at adjacent requests. For CFQ it's even worse, since there's no per-queue big rbtree for sorting.

Which is why I suggest special casing: Only invoke the expensive overlap checking if one of the requests is a discard. Otherwise use the standard path for writes.

James

--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>