

Re: ext3 allocate-with-reservation latencies

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-04/3052.html>

From: Stephen C. Tweedie (sct_at_redhat.com)

Date: 04/11/05

To: Mingming Cao <cmm@us.ibm.com>

Date: Mon, 11 Apr 2005 20:57:15 +0100

Hi,

On Mon, 2005-04-11 at 19:38, Mingming Cao wrote:

> *I agree. We should not skip the home block group of the file. I guess
> what I was suggesting is, if allocation from the home group failed and
> we continuing the linear search the rest of block groups, we could
> probably try to skip the block groups without enough usable free blocks
> to make a reservation.*

Fair enough. Once those are the only bgs left, performance is going to drop pretty quickly, but that's not really avoidable on a very full filesystem.

> *Ah.. I see the win with the read lock now: once the a reservation window
> is added, updating it (either winding it forward and or searching for a
> available window) probably is the majorirty of the operations on the
> tree, and using read lock for that should help reduce the latency.*

Right. The down side is that for things like a kernel "tar xf", we'll be doing lots of small file unpacks, and hopefully most files will be just one or two reservations — so there's little winding forward going on. The searching will still be improved in that case.

Note that this may improve average case latencies, but it's not likely to improve worst-case ones. We still need a write lock to install a new window, and that's going to have to wait for us to finish finding a free bit even if that operation starts using a read lock.

I'm not really sure what to do about worst-case here. For that, we really do want to drop the lock entirely while we do the bitmap scan.

That leaves two options. Hold a reservation while we do that; or don't. Holding one poses the problems we discussed before: either you hold a large reservation (bad for disk layout in the presence of concurrent allocators), or you hold smaller ones (high cost as you continually advance the window, which requires some read lock on the tree to avoid

bumping into the next window.)

Just how bad would it be if we didn't hold a lock _or_ a window at all while doing the search for new window space? I didn't like that alternative at first because of the problem when you've got multiple tasks trying to allocate in the same space at the same time; but given the locking overhead of the alternatives, I'm wondering if this is actually the lesser evil.

--Stephen

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>