

Re: [RFC] fsblock

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-06/msg11326.html>

- *From:* Chris Mason <chris.mason@xxxxxxxxxx>
 - *Date:* Wed, 27 Jun 2007 07:50:56 -0400
-

On Wed, Jun 27, 2007 at 07:32:45AM +0200, Nick Piggin wrote:

On Tue, Jun 26, 2007 at 08:34:49AM -0400, Chris Mason wrote:

On Tue, Jun 26, 2007 at 07:23:09PM +1000, David Chinner wrote:

On Tue, Jun 26, 2007 at 01:55:11PM +1000, Nick Piggin wrote:

[... fsblocks vs extent range mapping]

iomaps can double as range locks simply because iomaps are expressions of ranges within the file. Seeing as you can only access a given range exclusively to modify it, inserting an empty mapping into the tree as a range lock gives an effective method of allowing safe parallel reads, writes and allocation into the file.

The fsblocks and the vm page cache interface cannot be used to facilitate this because a radix tree is the wrong type of tree to store this information in. A sparse, range based tree (e.g. btree) is the right way to do this and it matches very well with a range based API.

I'm really not against the extent based page cache idea, but I kind of assumed it would be too big a change for this kind of generic setup. At any rate, if we'd like to do it, it may be best to ditch the idea of "attach mapping information to a page", and switch to "lookup mapping information and range locking for a page".

Well the get_block equivalent API is extent based one now, and I'll

Re: [RFC] fsblock

look at what is required in making map_fsblock a more generic call that could be used for an extent-based scheme.

An extent based thing IMO really isn't appropriate as the main generic layer here though. If it is really useful and popular, then it could be turned into generic code and sit along side fsblock or underneath fsblock...

Lets look at a typical example of how IO actually gets done today, starting with sys_write():

```
sys_write(file, buffer, 1MB)
for each page:
prepare_write()
allocate contiguous chunks of disk
attach buffers
copy_from_user()
commit_write()
dirty buffers
```

```
pdflush:
writepages()
find pages with contiguous chunks of disk
build and submit large bios
```

So, we replace prepare_write and commit_write with an extent based api, but we keep the dirty each buffer part. writepages has to turn that back into extents (bio sized), and the result is completely full of dark dark corner cases.

I do think fsblocks is a nice cleanup on its own, but Dave has a good point that it makes sense to look for ways generalize things even more.

-chris

-

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

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

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