Re: Unexpected splice "always copy" behavior observed

From: Linus Torvalds
Date: Wed May 19 2010 - 15:35:15 EST

- **Next message:** Stephane Eranian: "Re: [GIT PULL 0/2] perf session fix host_machine handling wrt build ids"
- **Previous message:** David Miller: "Re: [BUG] SLOB breaks Crypto"
- **In reply to:** Mathieu Desnoyers: "Re: Unexpected splice "always copy" behavior observed"
- **Next in thread:** Mathieu Desnoyers: "Re: Unexpected splice "always copy" behavior observed"
- **Messages sorted by:** [date] [thread] [subject] [author]

On Wed, 19 May 2010, Mathieu Desnoyers wrote:
>
> *Good point. This discard flag might do the trick and let us keep things simple.*
> *The major concern here is to keep the page cache disturbance relatively low.*
> *Which of new page allocation or stealing back the page has the lowest overhead*
> *would have to be determined with benchmarks.*

We could probably make it easier somehow to do the writeback and discard thing, but I have had _very_ good experiences with even a rather trivial file writer that basically used (iirc) 8MB windows, and the logic was very trivial:

- before writing a new 8M window, do "start writeback" (SYNC_FILE_RANGE_WRITE) on the previous window, and do a wait (SYNC_FILE_RANGE_WAIT_AFTER) on the window before that.

in fact, in its simplest form, you can do it like this (this is from my "overwrite disk images" program that I use on old disks):

```
for (index = 0; index < max_index ;index++) {
if (write(fd, buffer, BUFSIZE) != BUFSIZE)
break;
/* This won't block, but will start writeout asynchronously */
sync_file_range(fd, index*BUFSIZE, BUFSIZE, SYNC_FILE_RANGE_WRITE);
/* This does a blocking write-and-wait on any old ranges */
if (index)
sync_file_range(fd, (index-1)*BUFSIZE, BUFSIZE,
SYNC_FILE_RANGE_WAIT_BEFORE|SYNC_FILE_RANGE_WRITE|SYNC_FILE_RANGE_WAIT_AFTER);
}
```

and even if you don't actually do a discard (maybe we should add a SYNC_FILE_RANGE_DISCARD bit, right now you'd need to do a separate fadvise(FADV_DONTNEED) to throw it out) the system behavior is pretty nice, because the heavy writer gets good IO performance _and_ leaves only easy-to-free pages around after itself.

Linus

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

- **Next message:** Stephane Eranian: "Re: [GIT PULL 0/2] perf session fix host_machine handling wrt build ids"
- **Previous message:** David Miller: "Re: [BUG] SLOB breaks Crypto"
- **In reply to:** Mathieu Desnoyers: "Re: Unexpected splice "always copy" behavior observed"
- **Next in thread:** Mathieu Desnoyers: "Re: Unexpected splice "always copy" behavior observed"
- **Messages sorted by:** [date] [thread] [subject] [author]