

## Re: [DRIVER SUBMISSION] DRBD wants to go mainline

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-07/msg10115.html>

---

- *From:* Kyle Moffett <[mrlinuxman@xxxxxxxx](mailto:mrlinuxman@xxxxxxxx)>
  - *Date:* Mon, 23 Jul 2007 20:48:03 -0400
- 

For the guys on netdev, would you please look at the tcp\_recvmsg- threading and TCP\_NAGLE\_CORK issues below and give opinions on the best way to proceed?

One thing to remember, you don't necessarily have to merge every feature right away. As long as the new code is configured "off" by default with an "(EXPERIMENTAL)" warning, you can start getting the core parts and the cleanups upstream before you have to resolve all the issues with low-latency, dynamic-tracing-frameworks, etc.

On Jul 23, 2007, at 09:32:02, Lars Ellenberg wrote:

On Sun, Jul 22, 2007 at 09:32:02PM -0400, Kyle Moffett wrote:

```
+/* I don't remember why XCPU ...
+ * This is used to wake the asender,
+ * and to interrupt sending the sending task
+ * on disconnect.
+ */
+#define DRBD_SIG SIGXCPU
```

Don't use signals between kernel threads, use proper primitives like notifiers and waitqueues, which means you should also probably switch away from kernel\_thread() to the kthread\_\*() APIs. Also you should fix this FIXME or remove it if it no longer applies:-D.

right.

but how to I tell a network thread in tcp\_recvmsg to stop early, without using signals?

I'm not really a kernel-networking guy, so I can't answer this definitively, but I'm pretty sure the problem has been solved in many network filesystems and such, so I've added a netdev CC. The way I'd do it in userspace is with nonblocking IO and epoll(), that way I don't actually have to "stop" or "signal" the thread, I can just add a socket to epoll fd when I want to pay attention to it, and remove it from my epoll fd when I'm done with it. I'd assume there's some equivalent way in kernelspace based around the "struct kiocb \*iocb" and "int nonblock" parameters to the tcp\_recvmsg() kernel function.

Re: [DRIVER SUBMISSION] DRBD wants to go mainline

```
+/* see kernel/printk.c:printk_ratelimit
+ * macro, so it is easy do have independend rate limits at different locations
+ * "initializer element not constant ..." with kernel 2.4 :(
+ * so I initialize toks to something large
+ */
+#define DRBD_ratelimit(ratelimit_jiffies, ratelimit_burst) \
```

Any particular reason you can't just use `printk_ratelimit` for this?

I want to be able to do a rate-limit per specific message/code fragment, without affecting other messages or execution paths.

Ok, so could you change your patch to modify `__printk_ratelimit()` to also accept a "struct `printk_rate`" datastructure and make `printk_ratelimit()` call "`__printk_ratelimit(&global_printk_rate);`"??

Typically if `$KERNEL_FEATURE` is insufficient for your needs you should fix `$KERNEL_FEATURE` instead of duplicating a replacement in your driver. This applies to basically all of the things I'm talking about, kernel-threads, workqueues (BTW: I believe you can make your own custom workqueue thread(s) instead of using the default "events/" ones), debugging macros, fault-insertion, integer math, lock-checking, dynamic tracing, etc. If you find some reason that some generic code won't work for you, please try to fix it first so we can all benefit from it.

Umm, how about fixing this to actually use proper workqueues or something instead of this open-coded mess?

unlikely to happen "right now". but it is on our todo list...

Unfortunately problems like these need to be fixed before a mainline merge. Merging duplicated code is a big no-no, and historically there have been problems with people who merge code and never properly maintain it once it's in tree. As a result the rule is your code has to be easily maintainable before anybody will even \*consider\* merging it.

```
+/* I want the packet to fit within one page
+ * THINK maybe use a special bitmap header,
+ * including offset and compression scheme and whatnot
+ * Do not use PAGE_SIZE here! Use a architecture agnostic constant!
+ */
+#define BM_PACKET_WORDS ((4096-sizeof(struct
Drbd_Header))/sizeof (long))
```

Yuck. Definitely use `PAGE_SIZE` here, so at least if it's broken on an arch with multiple page sizes, somebody can grep for `PAGE_SIZE` to fix it. It also means that on archs/configs with 8k or 64k pages you won't waste a bunch of memory.

Re: [DRIVER SUBMISSION] DRBD wants to go mainline

No. This is not to allocate anything, but defines the chunk size with which we transmit the bitmap, when we have to. We need to be able to talk from one arch (say i586) to some other (say s390, or sparc, or whatever). The receiving side has a one-page buffer, from which it may or may not to endian-conversion. The hardcoded 4096 is the minimal common denominator here.

Ahhh. Please replace the constant "4096" with:

```
/* This is the maximum amount of bitmap we will send per packet */
# define MAX_BITMAP_CHUNK_SIZE 4096
# define BM_PACKET_WORDS \
((MAX_BITMAP_CHUNK_SIZE - sizeof(struct Drbd_Header))/sizeof(long))
```

It's more text but dramatically improves the readability by eliminating more magic numbers. This is a much milder case than I've seen in the past, so it's not that big of a deal.

```
+/* Dynamic tracing framework */
```

guess we have to explain first what this is for. it probably is not exactly what you think it is... but maybe I am just too ignorant about what you suggest here.

we'll have to defer discussion about this for when I'm done doing the trivial fix-ups, and provided an overall design overview.

From just poking at the code it looks vaguely similar to blktrace or something. We do have a lot of things in the kernel which fall under the name of "Dynamic tracing framework", so if you could look those over and see if you can't reuse them (or just remove the code for now and merge it later), that would be really useful.

```
+static inline void drbd_tcp_cork(struct socket *sock)
+{
+#if 1
+ mm_segment_t oldfs = get_fs();
+ int val = 1;
+
+ set_fs(KERNEL_DS);
+ tcp_setsockopt(sock->sk, SOL_TCP, TCP_CORK, (char *)&val,
sizeof(val) );
+ set_fs(oldfs);
+#else
+ tcp_sk(sock->sk)->nonagle |= TCP_NAGLE_CORK;
+#endif
+}
```

Yuck, why'd you do it this way? Probably because your tcp\_sk(sock->sk) stuff doesn't have proper locking, I'll bet. You can avoid all the extra wrapper

Re: [DRIVER SUBMISSION] DRBD wants to go mainline

crap by just looking in "do\_tcp\_setsockopt" and taking the appropriate lock:

```
static inline void drbd_tcp_cork(struct socket *sock)
{
    struct sock *sk = sock->sk;

    lock_sock(sk);
    tcp_sk(sk)->nonagle |= TCP_NAGLE_CORK;
    release_sock(sk);
}
```

it had performance improvements somewhen. I doubt it has still. maybe we just remove this.

NOTE: netdev guys, any chance you could comment on whether or not a RAID1-over-IP block device should be using TCP\_NAGLE\_CORK? If not, what should they be using instead?

```
+#define peer_mask role_mask
+#define pdsk_mask disk_mask
+#define susp_mask 1
+#define user_ismask 1
+#define aftr_ismask 1
+#define NS(T, S) \
+#define NS2(T1, S1, T2, S2) \
+#define NS3(T1, S1, T2, S2, T3, S3) \
+#define _NS(D, T, S) \
+#define _NS2(D, T1, S1, T2, S2) \
+#define _NS3(D, T1, S1, T2, S2, T3, S3) \
```

Grumble. When I earlier said I thought I was in macro hell, well, I was wrong. \*THIS\* is macro hell. What the fsck is that supposed to do? And it doesn't even include a \*SINGLE\* comment!!!

uhm. basically you are right. Phil will take over to explain why it is done that way...

Thanks, I'm looking forward to a good bout of hysterical laughter ;-)-D.

most other points I just snipped off of this mail will be handled as you suggested asap.

Great! Thanks. I'll take another deep dive into your git tree in a week or so when I've got more free time, but it would really help to have some smaller-ish patches which do any necessary preparatory cleanups. Good luck with your submission!

Cheers,

Re: [DRIVER SUBMISSION] DRBD wants to go mainline

Re: [DRIVER SUBMISSION] DRBD wants to go mainline

Kyle Moffett

—

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

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

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