

Re: duplicate read/write locks in net/pfil.c and netinet/ip_fw2.c

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2005-08/0178.html>

From: Stephan Uphoff (ups_at_tree.com)

Date: 08/21/05

To: John Baldwin <jhb@FreeBSD.org>

Date: Sat, 20 Aug 2005 19:02:23 -0400

On Thu, 2005-08-18 at 10:26, John Baldwin wrote:

> On Thursday 18 August 2005 10:18 am, Stephan Uphoff wrote:

>> On Thu, 2005-08-18 at 03:57, Luigi Rizzo wrote:

>>> On Thu, Aug 18, 2005 at 03:32:19AM +0200, Max Laier wrote:

>>>> On Thursday 18 August 2005 02:02, Luigi Rizzo wrote:

>>>

>>> ...

>>>

>>>> could you guys look at the following code and see if it makes sense,

>>>> or tell me where i am wrong ?

>>>>

>>>> It should solve the starvation and blocking trylock problems,

>>>> because the active reader does not hold the mutex in the critical

>>>

>>> ^^^^^

>>>

>>> i meant 'writer', sorry... as max said even in the current implementation

>>> the reader does not hold the lock.

>>>

>>>> int

>>>> RLOCK(struct rwlock *rwl, int try)

>>>> {

>>>> if (!try)

>>>> mtx_lock(&rwl->m);

>>>> else if (!mtx_trylock(&rwl->m))

>>>> return EBUSY;

>>>> if (rwl->writers == 0) /* no writer, pass */

>>>> rwl->readers++;

>>>> else {

>>>> rwl->br++;

>>>> cv_wait(&rwl->qr, &rwl->m);

>>>>

>>>> ^^^^^

>>>>

>>>> That we can't do. That's exactly the thing the existing sx(9)

> > > > *implementation does and where it breaks. The problem is that cv_wait()
> > > > is an implicit sleep which breaks when we try to RLOCK() with other
> > > > mutex already acquired.*
> > >
> > > *but that is not a solvable problem given that the *LOCK may be blocking.
> > > And the cv_wait is not an unconditioned sleep, it is one where you
> > > release the lock right before ans wait for an event to wake you up.
> > > In fact i don't understand why you consider spinning and sleeping
> > > on a mutex two different things.*
> >
> > *The major difference between sleeping (cv_wait,msleep,..) and blocking
> > on a mutex is priority inheritance.*
> > *If you need to be able to use (non-spin) mutexes while holding a
> > [R/W]LOCK and use a [R/W]LOCK while holding a (non-spin) mutex then you
> > need to implement priority inheritance for [R/W]LOCKS.*
> > *For the (single) write lock holder tracking priority is easy. (just like
> > a (non-spin) mutex). However priority inheritance to multiple readers is
> > more difficult as one needs to keep track of all holders of the lock.
> > Keeping track of all readers requires pre-allocated memory resources.
> > This memory could come from
> > 1) A limited global pool
> > 2) A limited per [R/W]LOCK pool
> > 3) A limited per thread pool
> > 4) As a parameter for acquiring a RLOCK
> > None of the choices are really pretty.
> > (1),(2) and (3) can lead to limiting reader parallelism when running out
> > of resources. (4) may be practically for some cases since the memory
> > could be allocated from stack. However since the memory must be valid
> > while holding a read lock choice (4) makes some algorithms (than use for
> > example lock crabbing) a bit harder to implement.*
>
> *Solaris handles the read case by only tracking the first thread to get a read
> lock (referred to as the "owner of record" IIRC) and only propagating
> priority to that thread and ignoring other readers. They admit it's not
> perfect as well. That's mentioned in the Solaris Internals book.*

Hi John,

If I recall correctly Solaris user threads get a better "system priority" when running/blocking in the kernel. This prevents threads holding critical kernel resources from being starved by user processes. I think this would also be a good idea for FreeBSD to prevent unbound priority inversions caused by vnode locks and other non-tracked resources. However even with these changes I don't think the Solaris implementation is the right way to go if we want to freely mix mutexes and [R/W]locks.

Stephan

freebsd-arch@freebsd.org mailing list

freebsd-arch: Re: duplicate read/write locks in net/pfil.c and netinet/ip_fw2.c

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@freebsd.org"