

## Re: CFR: New NFS Lock Manager

---

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/current/2008-03/msg00506.html>

---

- *From:* Erik Trulsson <ertr1013@xxxxxxxxxxxxxx>
  - *Date:* Fri, 21 Mar 2008 20:39:01 +0100
- 

On Fri, Mar 21, 2008 at 12:07:25PM -0700, Julian Elischer wrote:

Kip Macy wrote:

I think that for most of us this is the "nuclear reactor" in phk's bikeshed story :)

The Nuclear reactor is from the SVN guys talk on "poisonous personalities in open software projects" I think. where they declared it to be the logically opposite corollary of the Bikeshed.

No, it is much older than that, but is indeed the opposite of a bikeshed.

See the FreeBSD FAQ at

[http://www.freebsd.org/doc/en\\_US.ISO8859-1/books/faq/misc.html#BIKESHED-PAINTING](http://www.freebsd.org/doc/en_US.ISO8859-1/books/faq/misc.html#BIKESHED-PAINTING)  
for the whole story.

Doug, As a long term developer with a good track record, most of us are just going to say "probably works fine and I have no idea about that stuff" and leave it at that.

Yup. "nuclear reactor", indeed.

Who do you think has any idea about this stuff?

I had a tested and re-factored tcp\_output which got essentially no comments.

Re: CFR: New NFS Lock Manager

-Kip

On Fri, Mar 21, 2008 at 3:27 AM, Doug Rabson <dfr@xxxxxxxxxx> wrote:

As I mentioned previously, I have been working on a brand new NFS Lock Manager which runs in kernel mode and uses the normal local locking infrastructure for its state. I'm currently trying to tie up the last few loose ends before committing this work to current. You can find a snapshot of this code at <http://people.freebsd.org/~dfr/lockd-RC1-20032008.diff>

To try it out, take a recent current (I last merged with current on 20th March) and apply the patch. Build a kernel with the NFSLOCKD option and add '-k' to 'rpc\_lockd\_flags' in rc.conf. You will need to build and install at least a new libc and rpc.lockd.

At this point, it would be useful to get some extra eyes to look over my changes. In particular the following:

1. Choice of syscall number – I found one spare next to the NFS syscall and took that. The new syscall is listed in the FBSD\_1.1 namespace, possibly it should be somewhere else.
2. ABI compatibility – I extended the flock structure by one member (adding l\_sysid). I have added new operations to fcntl to support the new extended structure, leaving the old operations in place to work on the old structure. The kernel translates old to new and vice versa. No attempt is made to allow a new userland to work with an old kernel.
3. The local lock manager has had a complete rewrite to support required features. The new local lock manager supports a more flexible model of lock ownership (which can support remote lock

## Re: CFR: New NFS Lock Manager

owners). I have replaced the inadequate deadlock detection code with a new (and fast) graph based system. Using the deadlock graph, I was able to avoid the 'thundering herd' issues the old lock code had when many processes were contending for the same locked region. Given the extent of the changes, wider testing and review would be extremely welcome.

4. The NFS lock manager itself is brand new code and as such ought to be reviewed. I have also ported the userland sunrpc code to run in the kernel environment which may prove useful in future.

Highlights include:

- \* Thread-safe kernel RPC client – many threads can use the same RPC client handle safely with replies being de-multiplexed at the socket upcall (typically driven directly by the NIC interrupt) and handed off to whichever thread matches the reply. For UDP sockets, many RPC clients can share the same socket. This allows the use of a single privileged UDP port number to talk to an arbitrary number of remote hosts.

- \* Single-threaded kernel RPC server. Adding support for multi-threaded server would be relatively straightforward and would follow approximately the Solaris KPI. A single thread should be sufficient for the NLM since it should rarely block in normal operation.

- \* Kernel mode NLM server supporting cancel requests and granted callbacks. I've tested the NLM server reasonably extensively – it passes both my own tests and the NFS Connectathon locking tests running on Solaris, Mac OS X and Ubuntu Linux.

- \* Userland NLM client supported. While the NLM server doesn't have

Re: CFR: New NFS Lock Manager

support for the local NFS client's locking needs, it does have to field async replies and granted callbacks from remote NLMs that the local client has contacted. We relay these replies to the userland rpc.lockd over a local domain RPC socket.

\* IPv6 should be supported but has not been tested since I've been unable to get IPv6 to work properly with the Parallels virtual machines that I've been using for development.

\* Robust deadlock detection for the local lock manager. In particular it will detect deadlocks caused by a lock request that covers more than one blocking request. As required by the NLM protocol, all deadlock detection happens synchronously – a user is guaranteed that if a lock request isn't rejected immediately, the lock will eventually be granted. The old system allowed for a 'deferred deadlock' condition where a blocked lock request could wake up and find that some other deadlock-causing lock owner had beaten them to the lock.

\* Since both local and remote locks are managed by the same kernel locking code, local and remote processes can safely use file locks for mutual exclusion. Local processes have no fairness advantage compared to remote processes when contending to lock a region that has just been unlocked – the local lock manager enforces a strict first-come first-served model for both local and remote lockers.

—  
<Insert your favourite quote here.>  
Erik Trulsson  
ertr1013@xxxxxxxxxxxxxx

---

freebsd-current@xxxxxxxxxxxxx mailing list  
<http://lists.freebsd.org/mailman/listinfo/freebsd-current>  
To unsubscribe, send any mail to "freebsd-current-unsubscribe@xxxxxxxxxxxxx"