

Re: File trees: the deeper, the weirder

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/hackers/2006-10/msg00247.html>

- *From:* Kostik Belousov <kostikbel@xxxxxxxxxx>
 - *Date:* Mon, 30 Oct 2006 15:47:37 +0200
-

On Mon, Oct 30, 2006 at 04:05:19PM +0300, Yar Tikhiv wrote:

On Sun, Oct 29, 2006 at 11:32:58AM -0500, Matt Emmerton wrote:

[Restoring some OP context.]

On Sun, Oct 29, 2006 at 05:07:16PM +0300, Yar Tikhiv wrote:

As for the said program, it keeps its 1 Hz pace, mostly waiting on "vlruwk". It's killable, after a delay. The system doesn't show ...

Weird, eh? Any ideas what's going on?

I would guess that you need a new vnode to create the new file, but no vnodes are obvious candidates for freeing because they all have a child directory in use. Is there some sort of vnode clearing that goes on every second if we are short of vnodes?

See sys/vfs_subr.c, subroutine getnewvnode(). We call msleep() if we're waiting on vnodes to be created (or recycled). And just look at the 'hz' parameter passed to msleep()!

The calling process's mkdir() will end up waiting in getnewvnode() (in "vlruwk" state) while the vnru kernel thread does it's thing (which is to recycle vnodes.)

Either the vnru kernel thread has to work faster, or the caller has to sleep less, in order to avoid this lock-step behaviour.

Re: File trees: the deeper, the weirder

I'm afraid that, though your analysis is right, you arrive at wrong conclusions. The process waits for the whole second in `getnewvnode()` because the `vnlu` thread cannot free as much `vnodes` as it wants to. `vnlu_proc()` will wake up sleepers on `vnluproc_sig` (i.e., `getnewvnode()`) only if `(numvnodes <= desiredvnodes * 9 / 10)`. Whether this condition is attainable depends on `vlrureclaim()` (called from the `vnlu` thread) freeing `vnodes` at a sufficient rate. Perhaps `vlrureclaim()` just can't keep the pace at this conditions. `debug.vnlu_nowhere` increasing is an indication of that. Consequently, each `getnewvnode()` call sleeps 1 second, then grabs a `vnode` beyond `desiredvnodes`. It's no surprise that the 1 second delays start to appear after approx. `kern.maxvnodes` directories were created.

I think that David is right. The references `_from_` the directory make it immune to `vnode` reclamation. Try this patch. It is very unfair for `lsdf`.

Index: `sys/kern/vfs_subr.c`

```
=====  
RCS file: /usr/local/arch/ncvs/src/sys/kern/vfs_subr.c,v  
retrieving revision 1.685  
diff -u -r1.685 vfs_subr.c  
--- sys/kern/vfs_subr.c 2 Oct 2006 07:25:58 -0000 1.685  
+++ sys/kern/vfs_subr.c 30 Oct 2006 13:44:59 -0000  
@@ -582,7 +582,7 @@  
* If it's been deconstructed already, it's still  
* referenced, or it exceeds the trigger, skip it.  
*/  
- if (vp->v_usecount || !LIST_EMPTY(&(vp)->v_cache_src) ||  
+ if (vp->v_usecount || /* !LIST_EMPTY(&(vp)->v_cache_src) || */  
(vp->v_iflag & VI_DOOMED) != 0 || (vp->v_object != NULL &&  
vp->v_object->resident_page_count > trigger)) {  
VI_UNLOCK(vp);  
@@ -607,7 +607,7 @@  
* interlock, the other thread will be unable to drop the  
* vnode lock before our VOP_LOCK() call fails.  
*/  
- if (vp->v_usecount || !LIST_EMPTY(&(vp)->v_cache_src) ||  
+ if (vp->v_usecount || /* !LIST_EMPTY(&(vp)->v_cache_src) || */  
(vp->v_object != NULL &&  
vp->v_object->resident_page_count > trigger)) {  
VOP_UNLOCK(vp, LK_INTERLOCK, td);
```

Attachment: [pgpQH6iIs725h.pgp](#)

Description: PGP signature