

## Re: Packet loss every 30.999 seconds

---

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/stable/2007-12/msg00408.html>

---

- *From:* Bruce Evans <[brde@xxxxxxxxxxxxxxxxxx](mailto:brde@xxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 20 Dec 2007 01:15:40 +1100 (EST)
- 

On Tue, 18 Dec 2007, David G Lawrence wrote:

I got an almost identical delay (with 64000 vnodes).

Now, 17ms isn't much.

Says you. On modern systems, trying to run a pseudo real-time application on an otherwise quiescent system, 17ms is just short of an eternity. I agree that the syncer should be preemptable (which is what my bandaid patch attempts to do), but that probably wouldn't have helped my specific problem since my application was a user process, not a kernel thread.

FreeBSD isn't a real-time system, and 17ms isn't much for it. I saw lots of syscall delays of nearly 1 second while debugging this. (With another hat, I would say that 17 us was a long time in 1992. 17 us is hundreds of times longer now.)

One more followup (I swear I'm done, really!)... I have a laptop here that runs at 150MHz when it is in the lowest running CPU power save mode. At that speed, this bug causes a delay of more than 300ms and is enough to cause loss of keyboard input. I have to switch into high speed mode before I try to type anything, else I end up with random typos. Very annoying.

Yes, something is wrong if keystrokes are lost with CPUs that run at 150 kHz (sic) or faster.

Debugging shows that the problem is like I said. The loop really does take 125 ns per iteration. This time is actually not very much. The linked list of vnodes could hardly be designed better to maximize cache thrashing. My system has a fairly small L2 cache (512K or 1M), and even a few words from the vnode and the inode don't fit in the L2 cache when there are 64000 vnodes, but the vp and ip are also fairly well designed to maximize cache thrashing, so L2 cache thrashing starts

## Re: Packet loss every 30.999 seconds

at just a few thousand vnodes.

My system has fairly low latency main memory, else the problem would be larger:

```
% Memory latencies in nanoseconds – smaller is better
% (WARNING – may not be correct, check graphs)
% -----
% Host OS Mhz L1 $ L2 $ Main mem Guesses
% -----
% besplex.b FreeBSD 7.0-C 2205 1.361 5.6090 42.4 [PC3200 CL2.5 overclocked]
% sledge.fr FreeBSD 8.0-C 1802 1.666 8.9420 99.8
% freefall. FreeBSD 7.0-C 2778 0.746 6.6310 155.5
```

The loop makes the following memory accesses, at least in 5.2:

```
% loop:
% for (vp = TAILQ_FIRST(&mp->mnt_nvnodelist); vp != NULL; vp = nvp) {
% /*
% * If the vnode that we are about to sync is no longer
% * associated with this mount point, start over.
% */
% if (vp->v_mount != mp)
% goto loop;
% /*
% * Depend on the mntvnode_slock to keep things stable enough
% * for a quick test. Since there might be hundreds of
% * thousands of vnodes, we cannot afford even a subroutine
% * call unless there's a good chance that we have work to do.
% */
% nvp = TAILQ_NEXT(vp, v_nmntvnodes);
```

Access 1 word at vp offset 0x90. Costs 1 cache line. IIRC, my system has a cache line size of 0x40. Assume this, and that vp is aligned on a cache line boundary. So this access costs the cache line at vp offsets 0x80-0xbf.

```
% VI_LOCK(vp);
```

Access 1 word at vp offset 0x1c. Costs the cache line at vp offsets 0-0x3f.

```
% if (vp->v_iflag & VI_XLOCK) {
```

Access 1 word at vp offset 0x24. Cache hit.

```
% VI_UNLOCK(vp);
% continue;
% }
% ip = VTOI(vp);
```

Access 1 word at vp offset 0xa8. Cache hit.

Re: Packet loss every 30.999 seconds

Re: Packet loss every 30.999 seconds

```
% if (vp->v_type == VNON || ((ip->i_flag &
```

Access 1 word at vp offset 0xa0. Cache hit.

Access 1 word at ip offset 0x18. Assume that ip is aligned, as above. Costs the cache line at ip offsets 0-0x3f.

```
% (IN_ACCESS | IN_CHANGE | IN_MODIFIED | IN_UPDATE)) == 0 &&  
% TAILQ_EMPTY(&vp->v_dirtyblkhd)) {
```

Access 1 word at vp offset 0x48. Costs the cache line at vp offsets 0x40-0x7f.

```
% VI_UNLOCK(vp);
```

Reaccess 1 word at vp offset 0x1c. Cache hit.

```
% continue;  
% }
```

The total cost is 4 cache lines or 256 bytes per vnode. So with an L2 cache size of 1MB, the L2 cache will start thrashing at numvnodes = 4096. With thrashing, an at my main memory latency of 42.4 nsec, it might take  $4 * 42.4 = 169.6$  nsec to read main memory. This is similar to my observed time. Presumably things aren't quite that bad because there is some locality for the 3 lines in each vp. It might be possible to improve this a bit by accessing the lines sequentially and not interleaving the access to ip. Better, repack vp and move the IN\* flags from ip to vp (a change that has other advantages), so that everything is in 1 cache line per vp.

This isn't consistent with the delay increasing to 300 ms when the CPU is throttled -- memory shouldn't be throttled so much. On old machines, the memory was faster relative to the CPU, else noticeable 300[0] ms delays would have been common long ago. I think numvnodes grew large enough to bust L2 caches in the usual case even in 1992.

This code clearly wasn't designed with caches in mind :-). The cost of subroutine calls would be in the noise compared with the cost of cache misses.

Bruce

---

freebsd-stable@xxxxxxxxxxx mailing list

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

To unsubscribe, send any mail to "freebsd-stable-unsubscribe@xxxxxxxxxxx"

Re: Packet loss every 30.999 seconds