

Re: [RFC,PATCH] loopback: calls netif_receive_skb() instead of netif_rx()

Re: [RFC,PATCH] loopback: calls netif_receive_skb() instead of netif_rx()

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg11938.html>

- *From:* Ingo Molnar <mingo@xxxxxxx>
 - *Date:* Mon, 31 Mar 2008 12:44:03 +0200
-

* David Miller <davem@xxxxxxxxxxxxxx> wrote:

I don't think it's safe.

Every packet you receive can result in a sent packet, which in turn can result in a full packet receive path being taken, and yet again another sent packet.

And so on and so forth.

Some cases like this would be stack bugs, but wouldn't you like that bug to be a very busy cpu instead of a crash from overrunning the current stack?

sure.

But the core problem remains: our loopback networking scalability is poor. For plain localhost<->localhost connected sockets we hit the loopback device lock for every packet, and this very much shows up on real workloads on a quad already: the lock instruction in netif_rx is the most expensive instruction in a sysbench DB workload.

and it's not just about scalability, the plain algorithmic overhead is way too high as well:

```
$ taskset 1 ./bw_tcp -s
$ taskset 1 ./bw_tcp localhost
Socket bandwidth using localhost: 2607.09 MB/sec
$ taskset 1 ./bw_pipe
Pipe bandwidth: 3680.44 MB/sec
```

i dont think this is acceptable. Either we should fix loopback TCP performance or we should transparently switch to VFS pipes as a transport method when an app establishes a plain loopback connection (as long as there are no frills like content-modifying component in the

Re: [RFC,PATCH] loopback: calls netif_receive_skb() instead of netif_rx()

Re: [RFC,PATCH] loopback: calls netif_receive_skb() instead of netif_rx()

delivery path of packets after a connection has been established – which covers 99.9% of the real-life loopback cases).

I'm not suggesting we shouldn't use TCP for connection establishing – but if the TCP loopback packet transport is too slow we should use the VFS transport which is both more scalable, less cache-intense and has lower straight overhead as well.

Ingo

—

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

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

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