

poor network loopback performance and scalability (was: Re: [RFC,PATCH] loopback: calls netif_receive_skb() instead of netif_rx())

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg11947.html>

- *From:* Ingo Molnar <mingo@xxxxxxx>
 - *Date:* Mon, 31 Mar 2008 13:36:02 +0200
-

* David Miller <davem@xxxxxxxxxxxxxx> wrote:

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

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
```

Set your loopback MTU to some larger value if this result and the locking overhead upsets you.

yes, of course it "upsets me" – it shows up in macrobenchmarks as well (not just lmbench) – wouldnt (and shouldnt) that upset you?

And even with a ridiculously high MTU of 1048576 there's only a 13% improvement:

```
# ifconfig lo mtu 1048576
# taskset 1 ./bw_tcp -s
# taskset 1 ./bw_tcp localhost
Socket bandwidth using localhost: 2951.51 MB/sec
```

pipes are still another ~25% faster:

```
# taskset 1 ./bw_pipe
```

poor network loopback performance and scalability (was: Re: [RFC,PATCH] loopback: calls netif_receive_skb() instead of n

Pipe bandwidth: 3657.40 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 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 shouldnt 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.

[...]

Also, woe be to the application that wants fast local interprocess communication and doesn't use IPC_SHM, MAP_SHARED, pipes, or AF_UNIX sockets. (there's not just one better facility, there are four!)

From this perspective, people way-overemphasize loopback performance, and 999 times out of 1000 they prove their points using synthetic benchmarks.

And don't give me this garbage about the application wanting to be generic and therefore use IP sockets for everything. Either they want to be generic, or they want the absolute best performance. Trying to get an "or" and have both at the same time will result in ludicrous hacks ending up in the kernel.

i talked about the localhost data transport only (in the portion you dropped from your quotes), not about the connection API or the overall management of such sockets. There's absolutely no good technical reason i can see why plain loopback sockets should be forced to go over a global lock, or why apps should be forced to change to another API when the real problem is that kernel developers are lazy or incompetent to fix their code.

And i'm still trying to establish whether we have common ground for discussion: do you accept my numbers that TCP loopback transport performs badly when compared to pipes (i think you accepted that implicitly, but i dont want to put anything into your mouth).

Having agreed on that, do you share my view that it should be and could be fixed? Or do you claim that it cannot be fixed and wont ever be fixed?

Ingo

poor network loopback performance and scalability (was: Re: [RFC,PATCH] loopback: calls netif_receive_skb

poor network loopback performance and scalability (was: Re: [RFC,PATCH] loopback: calls netif_receive_skb() instead of n

--

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

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

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

poor network loopback performance and scalability (was: Re: [RFC,PATCH] loopback: calls netif_receive_skb()