

Re: sbwait state for loaded Apache server

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2003-07/0077.html>

From: Terry Lambert (tlambert2_at_mindspring.com)

Date: 07/23/03

Date: Tue, 22 Jul 2003 23:03:23 -0700
To: Michael Conlen <meconlen@obfuscated.net>

Michael Conlen wrote:

> *I'm working with an Apache webserver running 1400 apache processes and
> the system pusing somewhere in the area of 50-60Mbit/sec sustained. The
> system seems to top out around 60Mbit/sec and I see some minor
> degradation of server response times. The server response times are
> generally very very stable otherwise. Most of the apache processes are
> in the sbwait state. I've got 4Gig of memory, so I can play with some of
> the values (nmbclusters has been turned up and I never see delayed or
> dropped requests for mbufs).*
>
> *I don't see in my old Design & Implementation of 4.BSD (Red Book?) much
> about the state, and I don't a copy of TCP/IP Illustrated 2 handy these
> days, but if memory serves sbwait is waiting on a socket buffer
> resource. My guess is that these are processes waiting on the send
> buffer to drain.*
>
> *\$ netstat -an | egrep '[0-9] 3[0-9]{4}' | wc -l*
> *297*
>
> *seems to indicate that I've got a lot of processes waiting to drain.*
> *Looking at the actual output it shows most of these are ESTABLISHED.*

```
cd /usr/src/sys/kern
grep sbwait
```

The sleep call is the sbwait() function in uipc_socket2.c.

It's called:

- o On the send buffer from sendfile()
- o On the send buffer from sosend()
- o On the receive buffer from soreceive()

There's also a commented out call on the receive buffer in unp_gc(), which you can ignore, since it only deals with rights issues in the uipc cases related to AF_UNIX (UNIX domain sockets).

freebsd-performance: Re: sbwait state for loaded Apache server

The receive() case can probably be ignored, too, since it deals with blocking reads on sockets with no data present, and Apache generally doesn't do this.

So you are spending your time waiting for the send buffers to drain on an ssend() or a sendfile().

Basically, this probably means that you have a client on a slow link talking to your server on a fast link, or you have a client that is intentionally attempting to DOS you by sending a request and not keeping up the TCP/IP conversation, or you are running Microsoft's WAST HTTP benchmark program, and you really don't understand how it works, or you are running a Web Avalanche(tm) box against your server, or you have legitimate client traffic, but the client has dropped off the net.

If I had to guess, I'd say "slow clients".

- > *So my thought is by increasing the send queue size I could reduce this.*
- > *I've got a pretty good idea on the size of the files being sent and my*
- > *thoughts were to increase the send-q size to where Apache can write()*
- > *the file and go to the keep alive state quickly instead of waiting.*

This would most likely be an incredibly bad idea, since you will be more likely to go to FIN-WAIT-2 state, instead, and if you did go into an application level KeepAlive in Apache, you aren't going to be sending any KeepAlive messages that are going anywhere until the rest of the data has drained.

Further, you really want to delay closes until the client has done the close (1-2 seconds after you would have closed the socket) so that the client goes into FIN-WAIT-2 instead of you going into that state (servers don't want to be the ones to close the connection, because the FIN-WAIT-1 -> FIN-WAIT-2 transition doesn't get reversed ...even though you could technically pretend you never got the second FIN and solicit either an RST or another FIN... or no response, after a couple of which you could safely drop the connection.

- > *So the questions are*
- >
- > *Would this affect actual network performance*

Yes. It would negatively impact the overall total number of connections you could handle without running out of RAM, and it would penalize faster connections (people who get what they want and get the heck off your server, thus reducing your load) in favor of people with slower connections (people who get on your server and stay there forever, consuming your resources).

- > *Would this reduce load on the machine (a handy thing to do, but secondary)*
- > *given c = number of connections and q = queue adjustment and s = size of*

Re: sbwait state for loaded Apache server

freebsd-performance: Re: sbwait state for loaded Apache server

> *mbuf* do I just need to make sure I have $(c*q)/s$ buffers available, and
> any fudge?

Not really. All it would do is move your Apache processes into trying to poll clients who aren't going to answer, trying to ask them if they are still there and still need the connection.

Or it will drop you into FIN-WAIT-2, and if the client drops off the net, or is trying to DOS your server, leave it in that state for about 4 hours, chewing up those resources (the default for the timer that the TCP/IP standard doesn't permit people to implement to reap hanging FIN-WAIT-2's, but which people implement anyway). Much better to hack the stack to pretend it didn't get the second FIN at this point, send a FIN/ACK, and then only keep the connection around if you get a FIN back.

> *How do I know when I need to increase the overall system buffer size beyond 200 MB?*

That's a hard one to answer. The general answer is "When a overall system buffer size less than or equal to 200 MB is constraining my ability to service connections".

-- Terry

freebsd-performance@freebsd.org mailing list

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

To unsubscribe, send any mail to "freebsd-performance-unsubscribe@freebsd.org"