
Subject: Re: Problem with TCP window too large for TCPRCVBUF still present
Posted by [dev](#) on Tue, 05 Feb 2008 08:32:46 GMT

[View Forum Message](#) <> [Reply to Message](#)

try this one attached plz.

Marcin Owsiany wrote:

> Hi Vitaliy,

>

> [Cc'ing to Kirill, who sent the original patch]

>

> On Tue, Jan 08, 2008 at 11:12:49AM +0300, Vitaliy Gusev wrote:

>> On 11 October 2007 21:46:05 Marcin Owsiany wrote:

>>> Trying to debug a problem with stalling connections I found this thread:

>>> <http://forum.openvz.org/index.php?t=msg&goto=9018>

>>> which describes the exact problems I'm still having with 2.6.18-028.18

>>>

>>> Has it been fixed in this version? Or do you still need someone to test

>>> this patch?

>> Yes, this patch still requires testing.

>

> Can I have the patch as an attachment? The forum mangled the formatting

> and patch refuses to apply it..

>

--- ./diff.ve7777 2008-02-05 11:25:37.000000000 +0300

+++ ./diff 2008-02-05 11:32:56.000000000 +0300

@@ -1,131 +0,0 @@

-diff --git a/include/net/tcp.h b/include/net/tcp.h

-index 0ff49a5..7e8f200 100644

---- a/include/net/tcp.h

-+++ b/include/net/tcp.h

-@@@ -1815,8 +1815,9 @@@ static inline int tcp_win_from_space(int

-/* Note: caller must be prepared to deal with negative returns */

-static inline int tcp_space(const struct sock *sk)

-{

-- return tcp_win_from_space(sk->sk_rcvbuf -

-- atomic_read(&sk->sk_rmem_alloc));

-+ int ub_tcp_rcvbuf = (int) sock_bc(sk)->ub_tcp_rcvbuf;

-+ return tcp_win_from_space(min(sk->sk_rcvbuf, ub_tcp_rcvbuf)

-+ - atomic_read(&sk->sk_rmem_alloc));

-}

-

-static inline int tcp_full_space(const struct sock *sk)

-diff --git a/include/ub/beancounter.h b/include/ub/beancounter.h

-index 3d87afa..fc236e8 100644

---- a/include/ub/beancounter.h

-+++ b/include/ub/beancounter.h

```

-@@@ -144,6 +144,8 @@@ struct sock_private {
-unsigned long ubp_rmem_thres;
-unsigned long ubp_wmem_pressure;
-unsigned long ubp_maxadvms;
-+ /* Total size of all advertised receive windows for all tcp sockets */
-+ unsigned long ubp_rcv_wnd;
-unsigned long ubp_rmem_pressure;
-#define UB_RMEM_EXPAND 0
-#define UB_RMEM_KEEP 1
-@@@ -177,6 +179,7 @@@ #define ub_held_pages ppriv.ubp_held_pa
-struct sock_private spriv;
-#define ub_rmem_thres spriv.ubp_rmem_thres
-#define ub_maxadvms spriv.ubp_maxadvms
-+ #define ub_rcv_wnd spriv.ubp_rcv_wnd
-#define ub_rmem_pressure spriv.ubp_rmem_pressure
-#define ub_wmem_pressure spriv.ubp_wmem_pressure
-#define ub_tcp_sk_list spriv.ubp_tcp_socks
-diff --git a/include/ub/ub_sk.h b/include/ub/ub_sk.h
-index e65c9ed..02d0137 100644
---- a/include/ub/ub_sk.h
-+++ b/include/ub/ub_sk.h
-@@@ -34,6 +34,8 @@@ struct sock_beancounter {
-*/
-unsigned long poll_reserv;
-unsigned long forw_space;
-+ unsigned long ub_tcp_rcvbuf;
-+ unsigned long ub_rcv_wnd_old;
-/* fields below are protected by bc spinlock */
-unsigned long ub_waitspc; /* space waiting for */
-unsigned long ub_wcharged;
-diff --git a/kernel/ub/ub_net.c b/kernel/ub/ub_net.c
-index 74d651a..afee710 100644
---- a/kernel/ub/ub_net.c
-+++ b/kernel/ub/ub_net.c
-@@@ -420,6 +420,7 @@@ static int __sock_charge(struct sock *sk
-
-added_reserv = 0;
-added_forw = 0;
-+ skbc->ub_rcv_wnd_old = 0;
-if (res == UB_NUMTCPSOCK) {
-added_reserv = skb_charge_size(MAX_TCP_HEADER +
-1500 - sizeof(struct iphdr) -
-@@@ -439,6 +440,7 @@@ static int __sock_charge(struct sock *sk
-added_forw = 0;
-}
-skbc->forw_space = added_forw;
-+ skbc->ub_tcp_rcvbuf = added_forw + SK_STREAM_MEM_QUANTUM;
-}

```

```

-spin_unlock_irqrestore(&ub->ub_lock, flags);
-
-@@ -528,6 +530,7 @@ void ub_sock_uncharge(struct sock *sk)
-skbc->ub_wcharged, skbc->ub, skbc->ub->ub_uid);
-skbc->poll_reserv = 0;
-skbc->forw_space = 0;
-+ ub->ub_rcv_wnd -= is_tcp_sock ? tcp_sk(sk)->rcv_wnd : 0;
-spin_unlock_irqrestore(&ub->ub_lock, flags);
-
-uncharge_beancounter_notop(skbc->ub,
-@@ -768,6 +771,44 @@ static void ub_sockrcvbuf_uncharge(struc
-* UB_TCPRCVBUF
-*/
-
-+/*
-+ * UBC TCP window management mechanism.
-+ * Every socket may consume no more than sock_quantum.
-+ * sock_quantum depends on space available and ub_parms[UB_NUMTCPSOCK].held.
-+ */
-+static void ub_sock_tcp_update_rcvbuf(struct user_beancounter *ub,
-+ struct sock *sk)
-+{
-+ unsigned long allowed;
-+ unsigned long reserved;
-+ unsigned long available;
-+ unsigned long sock_quantum;
-+ struct tcp_opt *tp = tcp_sk(sk);
-+ struct sock_beancounter *skbc;
-+ skbc = sock_bc(sk);
-+
-+ if( ub->ub_parms[UB_NUMTCPSOCK].limit * ub->ub_maxadvms
-+ > ub->ub_parms[UB_TCPRCVBUF].limit) {
-+ /* this is defenitly shouldn't happend */
-+ return;
-+ }
-+ allowed = ub->ub_parms[UB_TCPRCVBUF].barrier;
-+ ub->ub_rcv_wnd += (tp->rcv_wnd - skbc->ub_rcv_wnd_old);
-+ skbc->ub_rcv_wnd_old = tp->rcv_wnd;
-+ reserved = ub->ub_parms[UB_TCPRCVBUF].held + ub->ub_rcv_wnd;
-+ available = (allowed < reserved)?
-+ 0:allowed - reserved;
-+ sock_quantum = max(allowed / ub->ub_parms[UB_NUMTCPSOCK].held,
-+ ub->ub_maxadvms);
-+ if ( skbc->ub_tcp_rcvbuf > sock_quantum) {
-+ skbc->ub_tcp_rcvbuf = sock_quantum;
-+ } else {
-+ skbc->ub_tcp_rcvbuf += min(sock_quantum - skbc->ub_tcp_rcvbuf,
-+ available);

```

```

- + }
- +
- +}
- +
- int ub_sock_tcp_chargerecv(struct sock *sk, struct sk_buff *skb,
-     enum ub_severity strict)
- {
-     @@ -804,6 +845,7 @@ int ub_sock_tcp_chargerecv(struct sock *
-     retval = 0;
-     for (ub = sock_bc(sk)->ub; ub->parent != NULL; ub = ub->parent);
-     spin_lock_irqsave(&ub->ub_lock, flags);
-     + ub_sock_tcp_update_rcvbuf(ub, sk);
-     ub->ub_parms[UB_TCPRCVBUF].held += chargesize;
-     if (ub->ub_parms[UB_TCPRCVBUF].held >
-         ub->ub_parms[UB_TCPRCVBUF].barrier &&
--- ./include/net/tcp.h.ve7777 2007-12-28 18:24:57.000000000 +0300
+++ ./include/net/tcp.h 2008-02-05 11:32:48.000000000 +0300
@@ -971,8 +971,9 @@ static inline int tcp_win_from_space(int
/* Note: caller must be prepared to deal with negative returns */
static inline int tcp_space(const struct sock *sk)
{
- return tcp_win_from_space(sk->sk_rcvbuf -
-     atomic_read(&sk->sk_rmem_alloc));
+ int ub_tcp_rcvbuf = (int) sock_bc(sk)->ub_tcp_rcvbuf;
+ return tcp_win_from_space(min(sk->sk_rcvbuf, ub_tcp_rcvbuf)
+     - atomic_read(&sk->sk_rmem_alloc));
}

static inline int tcp_full_space(const struct sock *sk)
--- ./include/ub/beancounter.h.ve7777 2007-12-28 18:24:56.000000000 +0300
+++ ./include/ub/beancounter.h 2008-02-05 11:32:48.000000000 +0300
@@ -151,6 +151,8 @@ struct sock_private {
    unsigned long ubp_rmem_thres;
    unsigned long ubp_wmem_pressure;
    unsigned long ubp_maxadvms;
+ /* Total size of all advertised receive windows for all tcp sockets */
+ unsigned long ubp_rcv_wnd;
    unsigned long ubp_rmem_pressure;
    int ubp_tw_count;
#define UB_RMEM_EXPAND      0
@@ -218,6 +220,7 @@ struct user_beancounter
    struct sock_private spriv;
#define ub_rmem_thres spriv.ubp_rmem_thres
#define ub_maxadvms spriv.ubp_maxadvms
+#define ub_rcv_wnd spriv.ubp_rcv_wnd
#define ub_rmem_pressure spriv.ubp_rmem_pressure
#define ub_wmem_pressure spriv.ubp_wmem_pressure
#define ub_tcp_sk_list spriv.ubp_tcp_socks

```

```

--- ./include/ub/ub_sk.h.ve7777 2007-12-28 18:24:52.000000000 +0300
+++ ./include/ub/ub_sk.h 2008-02-05 11:32:48.000000000 +0300
@@ -34,6 +34,8 @@ struct sock_beancounter {
    */
    unsigned long      poll_reserv;
    unsigned long      forw_space;
+ unsigned long      ub_tcp_rcvbuf;
+ unsigned long      ub_rcv_wnd_old;
    /* fields below are protected by bc spinlock */
    unsigned long      ub_waitspc;    /* space waiting for */
    unsigned long      ub_wcharged;
--- ./kernel/ub/ub_net.c.ve7777 2007-12-28 18:24:54.000000000 +0300
+++ ./kernel/ub/ub_net.c 2008-02-05 11:32:48.000000000 +0300
@@ -424,6 +424,7 @@ static int __sock_charge(struct sock *sk

    added_reserv = 0;
    added_forw = 0;
+ skbc->ub_rcv_wnd_old = 0;
    if (res == UB_NUMTCPSOCK) {
        added_reserv = skb_charge_size(MAX_TCP_HEADER +
            1500 - sizeof(struct iphdr) -
@@ -443,6 +444,7 @@ static int __sock_charge(struct sock *sk
        added_forw = 0;
    }
    skbc->forw_space = added_forw;
+ skbc->ub_tcp_rcvbuf = added_forw + SK_STREAM_MEM_QUANTUM;
    }
    spin_unlock_irqrestore(&ub->ub_lock, flags);

@@ -533,6 +535,7 @@ void ub_sock_uncharge(struct sock *sk)
    skbc->ub_wcharged, skbc->ub, skbc->ub->ub_uid);
    skbc->poll_reserv = 0;
    skbc->forw_space = 0;
+ ub->ub_rcv_wnd -= is_tcp_sock ? tcp_sk(sk)->rcv_wnd : 0;
    spin_unlock_irqrestore(&ub->ub_lock, flags);

    uncharge_beancounter_notop(skbc->ub,
@@ -793,6 +796,44 @@ static void ub_sockrcvbuf_uncharge(struc
    * UB_TCPRCVBUF
    */

+/*
+ * UBC TCP window management mechanism.
+ * Every socket may consume no more than sock_quantum.
+ * sock_quantum depends on space available and ub_parms[UB_NUMTCPSOCK].held.
+ */
+static void ub_sock_tcp_update_rcvbuf(struct user_beancounter *ub,
+ struct sock *sk)

```

```

+{
+ unsigned long allowed;
+ unsigned long reserved;
+ unsigned long available;
+ unsigned long sock_quantum;
+ struct tcp_opt *tp = tcp_sk(sk);
+ struct sock_beancounter *skbc;
+ skbc = sock_bc(sk);
+
+ if( ub->ub_parms[UB_NUMTCPSOCK].limit * ub->ub_maxadvms
+ > ub->ub_parms[UB_TCPRCVBUF].limit) {
+ /* this is defenitly shouldn't happend */
+ return;
+ }
+ allowed = ub->ub_parms[UB_TCPRCVBUF].barrier;
+ ub->ub_rcv_wnd += (tp->rcv_wnd - skbc->ub_rcv_wnd_old);
+ skbc->ub_rcv_wnd_old = tp->rcv_wnd;
+ reserved = ub->ub_parms[UB_TCPRCVBUF].held + ub->ub_rcv_wnd;
+ available = (allowed < reserved)?
+ 0:allowed - reserved;
+ sock_quantum = max(allowed / ub->ub_parms[UB_NUMTCPSOCK].held,
+ ub->ub_maxadvms);
+ if ( skbc->ub_tcp_rcvbuf > sock_quantum) {
+ skbc->ub_tcp_rcvbuf = sock_quantum;
+ } else {
+ skbc->ub_tcp_rcvbuf += min(sock_quantum - skbc->ub_tcp_rcvbuf,
+ available);
+ }
+
+}
+
+int ub_sock_tcp_chargerecv(struct sock *sk, struct sk_buff *skb,
+ enum ub_severity strict)
+{
@@ -829,6 +870,7 @@ int ub_sock_tcp_chargerecv(struct sock *
+   retval = 0;
+   ub = top_beancounter(sock_bc(sk)->ub);
+   spin_lock_irqsave(&ub->ub_lock, flags);
+ ub_sock_tcp_update_rcvbuf(ub, sk);
+ ub->ub_parms[UB_TCPRCVBUF].held += chargesize;
+ if (ub->ub_parms[UB_TCPRCVBUF].held >
+     ub->ub_parms[UB_TCPRCVBUF].barrier &&

```
