
Subject: [PATCH 2/6][INET]: Consolidate inet(6)_hash_connect.
Posted by [Pavel Emelianov](#) on Thu, 31 Jan 2008 12:32:09 GMT
[View Forum Message](#) <> [Reply to Message](#)

These two functions are the same except for what they call to "check_established" and "hash" for a socket.

This saves half-a-kilo for ipv4 and ipv6.

```
add/remove: 1/0 grow/shrink: 1/4 up/down: 582/-1128 (-546)
function          old  new  delta
__inet_hash_connect      -  577 +577
arp_ignore              108  113  +5
static.hint              8    4   -4
rt_worker_func          376  372  -4
inet6_hash_connect       584   25 -559
inet_hash_connect        586   25 -561
```

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
---
include/net/inet_hashtables.h | 5 ++
net/ipv4/inet_hashtables.c   | 32 ++++++++-----
net/ipv6/inet6_hashtables.c  | 93 +-----
3 files changed, 28 insertions(+), 102 deletions(-)
```

```
diff --git a/include/net/inet_hashtables.h b/include/net/inet_hashtables.h
index 761bdc0..a34a8f2 100644
--- a/include/net/inet_hashtables.h
+++ b/include/net/inet_hashtables.h
@@ -413,6 +413,11 @@ static inline struct sock *inet_lookup(struct inet_hashinfo *hashinfo,
    return sk;
}
```

```
+extern int __inet_hash_connect(struct inet_timewait_death_row *death_row,
+ struct sock *sk,
+ int (*check_established)(struct inet_timewait_death_row *,
+ struct sock *, __u16, struct inet_timewait_sock **),
+ void (*hash)(struct inet_hashinfo *, struct sock *));
extern int inet_hash_connect(struct inet_timewait_death_row *death_row,
    struct sock *sk);
#endif /* _INET_HASHTABLES_H */
```

```
diff --git a/net/ipv4/inet_hashtables.c b/net/ipv4/inet_hashtables.c
index 619c63c..b93d40f 100644
--- a/net/ipv4/inet_hashtables.c
+++ b/net/ipv4/inet_hashtables.c
@@ -348,11 +348,11 @@ void __inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk)
}
```

```
EXPORT_SYMBOL_GPL(__inet_hash);
```

```
/*
 * Bind a port for a connect operation and hash it.
 */
int inet_hash_connect(struct inet_timewait_death_row *death_row,
    struct sock *sk)
+int __inet_hash_connect(struct inet_timewait_death_row *death_row,
+ struct sock *sk,
+ int (*check_established)(struct inet_timewait_death_row *,
+ struct sock *, __u16, struct inet_timewait_sock **),
+ void (*hash)(struct inet_hashinfo *, struct sock *))
{
    struct inet_hashinfo *hinfo = death_row->hashinfo;
    const unsigned short snum = inet_sk(sk)->num;
@@ -385,9 +385,8 @@ int inet_hash_connect(struct inet_timewait_death_row *death_row,
    BUG_TRAP(!hlist_empty(&tb->owners));
    if (tb->fastreuse >= 0)
        goto next_port;
-    if (!__inet_check_established(death_row,
-        sk, port,
-        &tw))
+    if (!check_established(death_row, sk,
+        port, &tw))
        goto ok;
    goto next_port;
}
@@ -415,7 +414,7 @@ ok:
    inet_bind_hash(sk, tb, port);
    if (sk_unhashed(sk)) {
        inet_sk(sk)->sport = htons(port);
-    __inet_hash_nolisten(hinfo, sk);
+    hash(hinfo, sk);
    }
    spin_unlock(&head->lock);

@@ -432,17 +431,28 @@ ok:
    tb = inet_csk(sk)->icsk_bind_hash;
    spin_lock_bh(&head->lock);
    if (sk_head(&tb->owners) == sk && !sk->sk_bind_node.next) {
-    __inet_hash_nolisten(hinfo, sk);
+    hash(hinfo, sk);
    spin_unlock_bh(&head->lock);
    return 0;
    } else {
        spin_unlock(&head->lock);
        /* No definite answer... Walk to established hash table */
-    ret = __inet_check_established(death_row, sk, snum, NULL);
```

```

+ ret = check_established(death_row, sk, snum, NULL);
out:
    local_bh_enable();
    return ret;
}
}
+EXPORT_SYMBOL_GPL(__inet_hash_connect);
+
+/*
+ * Bind a port for a connect operation and hash it.
+ */
+int inet_hash_connect(struct inet_timewait_death_row *death_row,
+    struct sock *sk)
+{
+ return __inet_hash_connect(death_row, sk,
+    __inet_check_established, __inet_hash_nolisten);
+}

EXPORT_SYMBOL_GPL(inet_hash_connect);
diff --git a/net/ipv6/inet6_hashtables.c b/net/ipv6/inet6_hashtables.c
index 06b01be..ece6d0e 100644
--- a/net/ipv6/inet6_hashtables.c
+++ b/net/ipv6/inet6_hashtables.c
@@ -233,97 +233,8 @@ static inline u32 inet6_sk_port_offset(const struct sock *sk)
int inet6_hash_connect(struct inet_timewait_death_row *death_row,
    struct sock *sk)
{
- struct inet_hashinfo *hinfo = death_row->hashinfo;
- const unsigned short snum = inet_sk(sk)->num;
- struct inet_bind_hashbucket *head;
- struct inet_bind_bucket *tb;
- int ret;
-
- if (snum == 0) {
- int i, port, low, high, remaining;
- static u32 hint;
- const u32 offset = hint + inet6_sk_port_offset(sk);
- struct hlist_node *node;
- struct inet_timewait_sock *tw = NULL;
-
- inet_get_local_port_range(&low, &high);
- remaining = (high - low) + 1;
-
- local_bh_disable();
- for (i = 1; i <= remaining; i++) {
- port = low + (i + offset) % remaining;
- head = &hinfo->bhash[inet_bhashfn(port, hinfo->bhash_size)];
- spin_lock(&head->lock);

```

```

-
- /* Does not bother with rcv_saddr checks,
-  * because the established check is already
-  * unique enough.
-  */
- inet_bind_bucket_for_each(tb, node, &head->chain) {
-     if (tb->port == port) {
-         BUG_TRAP(!hlist_empty(&tb->owners));
-         if (tb->fastreuse >= 0)
-             goto next_port;
-         if (!__inet6_check_established(death_row,
-                                         sk, port,
-                                         &tw))
-             goto ok;
-         goto next_port;
-     }
- }
-
- tb = inet_bind_bucket_create(hinfo->bind_bucket_cache,
-                             head, port);
- if (!tb) {
-     spin_unlock(&head->lock);
-     break;
- }
- tb->fastreuse = -1;
- goto ok;
-
- next_port:
-     spin_unlock(&head->lock);
- }
- local_bh_enable();
-
- return -EADDRNOTAVAIL;
-
-ok:
- hint += i;
-
- /* Head lock still held and bh's disabled */
- inet_bind_hash(sk, tb, port);
- if (sk_unhashed(sk)) {
-     inet_sk(sk)->sport = htons(port);
-     __inet6_hash(hinfo, sk);
- }
- spin_unlock(&head->lock);
-
- if (tw) {
-     inet_twsk_deschedule(tw, death_row);
-     inet_twsk_put(tw);

```

```

- }
-
- ret = 0;
- goto out;
- }
-
- head = &hinfo->bhash[inet_bhashfn(snum, hinfo->bhash_size)];
- tb = inet_csk(sk)->icsk_bind_hash;
- spin_lock_bh(&head->lock);
-
- if (sk_head(&tb->owners) == sk && sk->sk_bind_node.next == NULL) {
- __inet6_hash(hinfo, sk);
- spin_unlock_bh(&head->lock);
- return 0;
- } else {
- spin_unlock(&head->lock);
- /* No definite answer... Walk to established hash table */
- ret = __inet6_check_established(death_row, sk, snum, NULL);
-out:
- local_bh_enable();
- return ret;
- }
+ return __inet_hash_connect(death_row, sk,
+ __inet6_check_established, __inet6_hash);
}

```

```
EXPORT_SYMBOL_GPL(inet6_hash_connect);
```

```
--
```

1.5.3.4
