
Subject: [PATCH net-2.6.25 (resend) 1/3] Uninline the __inet_hash function
Posted by [Pavel Emelianov](#) on Thu, 20 Dec 2007 09:46:07 GMT
[View Forum Message](#) <> [Reply to Message](#)

This one is used in quite many places in the networking code and seems to big to be inline.

After the patch net/ipv4/build-in.o loses ~650 bytes:
add/remove: 2/0 grow/shrink: 0/5 up/down: 461/-1114 (-653)

function	old	new	delta
__inet_hash_nolisten	-	282	+282
__inet_hash	-	179	+179
tcp_sacktag_write_queue	2255	2254	-1
__inet_lookup_listener	284	274	-10
tcp_v4_syn_recv_sock	755	493	-262
tcp_v4_hash	389	35	-354
inet_hash_connect	1086	599	-487

This version addresses the issue pointed by Eric, that while being inline this function was optimized by gcc in respect to the 'listen_possible' argument.

(Patches 2 and 3 in this series are still applied after this)

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

```
diff --git a/include/net/inet_hashtables.h b/include/net/inet_hashtables.h
index fef4442..65ddb25 100644
--- a/include/net/inet_hashtables.h
+++ b/include/net/inet_hashtables.h
@@ -264,37 +264,14 @@ static inline void inet_listen_unlock(struct inet_hashinfo *hashinfo)
     wake_up(&hashinfo->lhash_wait);
 }

-static inline void __inet_hash(struct inet_hashinfo *hashinfo,
-    struct sock *sk, const int listen_possible)
-{
-    struct hlist_head *list;
-    rwlock_t *lock;
-
-    BUG_TRAP(sk_unhashed(sk));
-    if (listen_possible && sk->sk_state == TCP_LISTEN) {
-        list = &hashinfo->listening_hash[inet_sk_listen_hashfn(sk)];
-        lock = &hashinfo->lhash_lock;
-        inet_listen_wlock(hashinfo);
-    } else {
```

```

- struct inet_eshash_bucket *head;
- sk->sk_hash = inet_sk_eshashfn(sk);
- head = inet_eshash_bucket(hashinfo, sk->sk_hash);
- list = &head->chain;
- lock = inet_eshash_lockp(hashinfo, sk->sk_hash);
- write_lock(lock);
- }
- __sk_add_node(sk, list);
- sock_prot_inc_use(sk->sk_prot);
- write_unlock(lock);
- if (listen_possible && sk->sk_state == TCP_LISTEN)
- wake_up(&hashinfo->lhash_wait);
- }
+extern void __inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk);
+extern void __inet_hash_nolisten(struct inet_hashinfo *hinfo, struct sock *sk);

static inline void inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk)
{
    if (sk->sk_state != TCP_CLOSE) {
        local_bh_disable();
- __inet_hash(hashinfo, sk, 1);
+ __inet_hash(hashinfo, sk);
        local_bh_enable();
    }
}

diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index 02fc91c..f450df2 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -408,7 +408,7 @@ struct sock *dccp_v4_request_rcv_sock(struct sock *sk, struct sk_buff
*skb,

    dccp_sync_mss(newsk, dst_mtu(dst));

- __inet_hash(&dccp_hashinfo, newsk, 0);
+ __inet_hash_nolisten(&dccp_hashinfo, newsk);
    __inet_inherit_port(&dccp_hashinfo, sk, newsk);

    return newsk;

diff --git a/net/ipv4/inet_hashtables.c b/net/ipv4/inet_hashtables.c
index b07e2d3..2e5814a 100644
--- a/net/ipv4/inet_hashtables.c
+++ b/net/ipv4/inet_hashtables.c
@@ -305,6 +305,48 @@ static inline u32 inet_sk_port_offset(const struct sock *sk)
    inet->dport);
}

+void __inet_hash_nolisten(struct inet_hashinfo *hashinfo, struct sock *sk)

```

```

+{
+ struct hlist_head *list;
+ rwlock_t *lock;
+ struct inet_eshash_bucket *head;
+
+ BUG_TRAP(sk_unhashed(sk));
+
+ sk->sk_hash = inet_sk_eshashfn(sk);
+ head = inet_eshash_bucket(hashinfo, sk->sk_hash);
+ list = &head->chain;
+ lock = inet_eshash_lockp(hashinfo, sk->sk_hash);
+
+ write_lock(lock);
+ __sk_add_node(sk, list);
+ sock_prot_inc_use(sk->sk_prot);
+ write_unlock(lock);
+}
+EXPORT_SYMBOL_GPL(__inet_hash_nolisten);
+
+void __inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk)
+{
+ struct hlist_head *list;
+ rwlock_t *lock;
+
+
+ if (sk->sk_state != TCP_LISTEN) {
+ __inet_hash_nolisten(hashinfo, sk);
+ return;
+ }
+
+ BUG_TRAP(sk_unhashed(sk));
+ list = &hashinfo->listening_hash[inet_sk_listen_hashfn(sk)];
+ lock = &hashinfo->lhash_lock;
+
+ inet_listen_wlock(hashinfo);
+ __sk_add_node(sk, list);
+ sock_prot_inc_use(sk->sk_prot);
+ write_unlock(lock);
+ wake_up(&hashinfo->lhash_wait);
+}
+EXPORT_SYMBOL_GPL(__inet_hash);
+
+/*
+ * Bind a port for a connect operation and hash it.
+ */
@@ -372,7 +414,7 @@ ok:
inet_bind_hash(sk, tb, port);
if (sk_unhashed(sk)) {
inet_sk(sk)->sport = htons(port);

```

```
- __inet_hash(hinfo, sk, 0);
+ __inet_hash_nolisten(hinfo, sk);
}
spin_unlock(&head->lock);
```

@@ -389,7 +431,7 @@ 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(hinfo, sk, 0);
+ __inet_hash_nolisten(hinfo, sk);
spin_unlock_bh(&head->lock);
return 0;
} else {
```

diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c

index 652c323..fc9bdd8 100644

--- a/net/ipv4/tcp_ipv4.c

+++ b/net/ipv4/tcp_ipv4.c

```
@@ -1478,7 +1478,7 @@ struct sock *tcp_v4_syn_recv_sock(struct sock *sk, struct sk_buff
*skb,
}
#endif
```

```
- __inet_hash(&tcp_hashinfo, newsk, 0);
+ __inet_hash_nolisten(&tcp_hashinfo, newsk);
__inet_inherit_port(&tcp_hashinfo, sk, newsk);

return newsk;
```
