
Subject: [PATCH 3/6][NETNS]: Make bind buckets live in net namespaces.

Posted by [Pavel Emelianov](#) on Thu, 31 Jan 2008 12:35:39 GMT

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

This tags the inet_bind_bucket struct with net pointer, initializes it during creation and makes a filtering during lookup.

A better hashfn, that takes the net into account is to be done in the future, but currently all bind buckets with similar port will be in one hash chain.

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

```
include/net/inet_hashtables.h | 2 ++
net/ipv4/inet_connection_sock.c | 8 +++++---
net/ipv4/inet_hashtables.c | 8 ++++++--
3 files changed, 13 insertions(+), 5 deletions(-)
```

```
diff --git a/include/net/inet_hashtables.h b/include/net/inet_hashtables.h
```

```
index a34a8f2..55532b9 100644
```

```
--- a/include/net/inet_hashtables.h
```

```
+++ b/include/net/inet_hashtables.h
```

```
@@ -74,6 +74,7 @@ struct inet_ehash_bucket {
 * ports are created in O(1) time? I thought so. ;-) -DaveM
 */
```

```
struct inet_bind_bucket {
```

```
+ struct net *ib_net;
```

```
    unsigned short port;
```

```
    signed short fastreuse;
```

```
    struct hlist_node node;
```

```
@@ -194,6 +195,7 @@ static inline void inet_ehash_locks_free(struct inet_hashinfo *hashinfo)
```

```
extern struct inet_bind_bucket *
```

```
    inet_bind_bucket_create(struct kmem_cache *cachep,
```

```
+ struct net *net,
```

```
    struct inet_bind_hashbucket *head,
```

```
    const unsigned short snum);
```

```
extern void inet_bind_bucket_destroy(struct kmem_cache *cachep,
```

```
diff --git a/net/ipv4/inet_connection_sock.c b/net/ipv4/inet_connection_sock.c
```

```
index 7801cce..de5a41d 100644
```

```
--- a/net/ipv4/inet_connection_sock.c
```

```
+++ b/net/ipv4/inet_connection_sock.c
```

```
@@ -87,6 +87,7 @@ int inet_csk_get_port(struct inet_hashinfo *hashinfo,
```

```
    struct hlist_node *node;
```

```
    struct inet_bind_bucket *tb;
```

```
    int ret;
```

```

+ struct net *net = sk->sk_net;

local_bh_disable();
if (!snum) {
@@ -100,7 +101,7 @@ int inet_csk_get_port(struct inet_hashinfo *hashinfo,
    head = &hashinfo->bhash[inet_bhashfn(rover, hashinfo->bhash_size)];
    spin_lock(&head->lock);
    inet_bind_bucket_for_each(tb, node, &head->chain)
-   if (tb->port == rover)
+   if (tb->ib_net == net && tb->port == rover)
        goto next;
    break;
next:
@@ -127,7 +128,7 @@ int inet_csk_get_port(struct inet_hashinfo *hashinfo,
    head = &hashinfo->bhash[inet_bhashfn(snum, hashinfo->bhash_size)];
    spin_lock(&head->lock);
    inet_bind_bucket_for_each(tb, node, &head->chain)
-   if (tb->port == snum)
+   if (tb->ib_net == net && tb->port == snum)
        goto tb_found;
}
tb = NULL;
@@ -147,7 +148,8 @@ tb_found:
}
tb_not_found:
ret = 1;
- if (!tb && (tb = inet_bind_bucket_create(hashinfo->bind_bucket_cachep, head, snum)) == NULL)
+ if (!tb && (tb = inet_bind_bucket_create(hashinfo->bind_bucket_cachep,
+ net, head, snum)) == NULL)
    goto fail_unlock;
if (hlist_empty(&tb->owners)) {
    if (sk->sk_reuse && sk->sk_state != TCP_LISTEN)
diff --git a/net/ipv4/inet_hashtables.c b/net/ipv4/inet_hashtables.c
index b93d40f..db1e53a 100644
--- a/net/ipv4/inet_hashtables.c
+++ b/net/ipv4/inet_hashtables.c
@@ -28,12 +28,14 @@
 * The bindhash mutex for snum's hash chain must be held here.
 */
struct inet_bind_bucket *inet_bind_bucket_create(struct kmem_cache *cachep,
+ struct net *net,
+ struct inet_bind_hashbucket *head,
+ const unsigned short snum)
{
    struct inet_bind_bucket *tb = kmem_cache_alloc(cachep, GFP_ATOMIC);

    if (tb != NULL) {
+   tb->ib_net = net;

```

```

tb->port    = snum;
tb->fastreuse = 0;
INIT_HLIST_HEAD(&tb->owners);
@@ -359,6 +361,7 @@ int __inet_hash_connect(struct inet_timewait_death_row *death_row,
    struct inet_bind_hashbucket *head;
    struct inet_bind_bucket *tb;
    int ret;
+ struct net *net = sk->sk_net;

    if (!snum) {
        int i, remaining, low, high, port;
@@ -381,7 +384,7 @@ int __inet_hash_connect(struct inet_timewait_death_row *death_row,
    * unique enough.
    */
    inet_bind_bucket_for_each(tb, node, &head->chain) {
- if (tb->port == port) {
+ if (tb->ib_net == net && tb->port == port) {
        BUG_TRAP(!hlist_empty(&tb->owners));
        if (tb->fastreuse >= 0)
            goto next_port;
@@ -392,7 +395,8 @@ int __inet_hash_connect(struct inet_timewait_death_row *death_row,
    }
}

- tb = inet_bind_bucket_create(hinfo->bind_bucket_cachep, head, port);
+ tb = inet_bind_bucket_create(hinfo->bind_bucket_cachep,
+ net, head, port);
    if (!tb) {
        spin_unlock(&head->lock);
        break;

```

--

1.5.3.4