
Subject: [PATCH] Consolidate udp hash calculations
Posted by [xemul](#) on Fri, 04 May 2007 14:47:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Make access to udphash/udplitehash symmetrical to inet hashes.

This may also help network namespaces, since they tend to use one hash for different namespaces by selecting the hash chain depending on a hash value and the namespace.

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

```
diff --git a/include/linux/udp.h b/include/linux/udp.h
index 6de445c..a40ccd4 100644
--- a/include/linux/udp.h
+++ b/include/linux/udp.h
@@ -49,6 +49,11 @@ static inline struct udphdr *udp_hdr(con
#include <net/inet_sock.h>
#define UDP_HTABLE_SIZE 128

+static inline unsigned int udp_hashfn(unsigned int num)
+{
+ return num & (UDP_HTABLE_SIZE - 1);
+}
+
struct udp_sock {
/* inet_sock has to be the first member */
struct inet_sock inet;
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 113e0c4..8ea1c81 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
@@ -132,7 +132,7 @@ static inline int __udp_lib_port_inuse(u
struct hlist_node *node;
struct inet_sock *inet;

- sk_for_each(sk, node, &udptable[hash & (UDP_HTABLE_SIZE - 1)]) {
+ sk_for_each(sk, node, &udptable[udp_hashfn(hash)]) {
if (sk->sk_hash != hash)
continue;
inet = inet_sk(sk);
@@ -159,7 +159,6 @@ int __udp_lib_get_port(struct sock *sk,
const struct sock *sk2 )
{
struct hlist_node *node;
- struct hlist_head *head;
```

```

struct sock *sk2;
unsigned int hash;
int error = 1;
@@ -175,10 +174,11 @@ int __udp_lib_get_port(struct sock *sk,
    best = result = *port_rover;
    for (i = 0; i < UDP_HTABLE_SIZE; i++, result++) {
        int size;
+   struct hlist_head *head;

        hash = hash_port_and_addr(result,
            inet_sk(sk)->rcv_saddr);
-   head = &udptable[hash & (UDP_HTABLE_SIZE - 1)];
+   head = &udptable[udp_hashfn(hash)];
        if (hlist_empty(head)) {
            if (result > sysctl_local_port_range[1])
                result = sysctl_local_port_range[0] +
@@ -222,9 +222,7 @@ gotit:
            *port_rover = snum = result;
        } else {
            hash = hash_port_and_addr(snum, 0);
-   head = &udptable[hash & (UDP_HTABLE_SIZE - 1)];
-
-   sk_for_each(sk2, node, head)
+   sk_for_each(sk2, node, &udptable[udp_hashfn(hash)])
        if (sk2->sk_hash == hash &&
            sk2 != sk &&
            inet_sk(sk2)->num == snum &&
@@ -237,9 +235,7 @@ gotit:
            if (inet_sk(sk)->rcv_saddr) {
                hash = hash_port_and_addr(snum,
                    inet_sk(sk)->rcv_saddr);
-   head = &udptable[hash & (UDP_HTABLE_SIZE - 1)];
-
-   sk_for_each(sk2, node, head)
+   sk_for_each(sk2, node, &udptable[udp_hashfn(hash)])
        if (sk2->sk_hash == hash &&
            sk2 != sk &&
            inet_sk(sk2)->num == snum &&
@@ -255,8 +251,7 @@ gotit:
            inet_sk(sk)->num = snum;
            sk->sk_hash = hash;
            if (sk_unhashed(sk)) {
-   head = &udptable[hash & (UDP_HTABLE_SIZE - 1)];
-   sk_add_node(sk, head);
+   sk_add_node(sk, &udptable[udp_hashfn(hash)]);
            sock_prot_inc_use(sk->sk_prot);
        }
        error = 0;

```

```
@@ -304,7 +299,7 @@ static struct sock *__udp4_lib_lookup(__
```

lookup:

```
- sk_for_each(sk, node, &udptable[hash & (UDP_HTABLE_SIZE - 1)]) {  
+ sk_for_each(sk, node, &udptable[udp_hashfn(hash)]) {  
    struct inet_sock *inet = inet_sk(sk);
```

```
    if (sk->sk_hash != hash || ipv6_only_sock(sk) ||  
@@ -1205,8 +1200,8 @@ static int __udp4_lib_mcast_deliver(stru
```

```
    read_lock(&udp_hash_lock);
```

```
- sk = sk_head(&udptable[hash & (UDP_HTABLE_SIZE - 1)]);  
- skw = sk_head(&udptable[hashwild & (UDP_HTABLE_SIZE - 1)]);  
+ sk = sk_head(&udptable[udp_hashfn(hash)]);  
+ skw = sk_head(&udptable[udp_hashfn(hashwild)]);
```

```
    sk = udp_v4_mcast_next(sk, hash, hport, daddr, uh->source, saddr, dif);  
    if (!sk) {  
diff --git a/net/ipv6/udp.c b/net/ipv6/udp.c  
index b083c09..a0a8915 100644
```

```
--- a/net/ipv6/udp.c  
+++ b/net/ipv6/udp.c
```

```
@@ -67,7 +67,7 @@ static struct sock *__udp6_lib_lookup(st  
    int badness = -1;
```

```
    read_lock(&udp_hash_lock);  
- sk_for_each(sk, node, &udptable[hnum & (UDP_HTABLE_SIZE - 1)]) {  
+ sk_for_each(sk, node, &udptable[udp_hashfn(hnum)]) {  
    struct inet_sock *inet = inet_sk(sk);
```

```
    if (sk->sk_hash == hnum && sk->sk_family == PF_INET6) {  
@@ -350,7 +350,7 @@ static int __udp6_lib_mcast_deliver(stru  
    int dif;
```

```
    read_lock(&udp_hash_lock);  
- sk = sk_head(&udptable[ntohs(uh->dest) & (UDP_HTABLE_SIZE - 1)]);  
+ sk = sk_head(&udptable[udp_hashfn(ntohs(uh->dest))]);  
    dif = inet6_iif(skb);  
    sk = udp_v6_mcast_next(sk, uh->dest, daddr, uh->source, saddr, dif);  
    if (!sk) {
```

Subject: Re: [PATCH] Consolidate udp hash calculations
Posted by [davem](#) on Fri, 04 May 2007 19:39:09 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Pavel Emelianov <xemul@sw.ru>

Date: Fri, 04 May 2007 18:51:36 +0400

> Make access to udphash/udplitehash symmetrical to inet hashes.

>

> This may also help network namespaces, since they tend to use

> one hash for different namespaces by selecting the hash chain

> depending on a hash value and the namespace.

>

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

This is not the "UDP Hash Function", it is a mask to bring the hash value modulo of the hash table size.

So your function name is misleading and makes the code even worse.

I'm also not so sure this cleanup really adds any clarity, even with a proper function name.

I can only assume you wish to do something with the UDP hash table sizes, and therefore only want to have a need to touch one function.

Subject: Re: [PATCH] Consolidate udp hash calculations

Posted by [xemul](#) on Mon, 07 May 2007 07:02:36 GMT

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

David Miller wrote:

> From: Pavel Emelianov <xemul@sw.ru>

> Date: Fri, 04 May 2007 18:51:36 +0400

>

>> Make access to udphash/udplitehash symmetrical to inet hashes.

>>

>> This may also help network namespaces, since they tend to use

>> one hash for different namespaces by selecting the hash chain

>> depending on a hash value and the namespace.

>>

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

>

> This is not the "UDP Hash Function", it is a mask to bring

> the hash value modulo of the hash table size.

>

> So your function name is misleading and makes the code

> even worse.

>

> I'm also not so sure this cleanup really adds any clarity,

> even with a proper function name.

The same is applicable to `inet_lhashfn` and `inet_bhashfn`, but they are called `hashfn-s` and do exist in kernel. This `hashfn` will make things look similar.

> I can only assume you wish to do something with the UDP hash table
> sizes, and therefore only want to have a need to touch one function.

You are right, I have told that this must help network namespaces, by making the hash differ depending on the namespace passed.

Subject: Re: [PATCH] Consolidate udp hash calculations

Posted by [davem](#) on Mon, 07 May 2007 07:04:47 GMT

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

From: Pavel Emelianov <xemul@sw.ru>

Date: Mon, 07 May 2007 11:07:15 +0400

> David Miller wrote:

> > I can only assume you wish to do something with the UDP hash table
> > sizes, and therefore only want to have a need to touch one function.

>

> You are right, I have told that this must help network namespaces,
> by making the hash differ depending on the namespace passed.

Then make your code rearrangements as part of your namespace patches so we can see what the consolidation is even for.
