

# [PATCH] vlan & net drivers: avoid a 4-order allocation

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg10039.html>

---

- *From:* Dan Aloni <da-x@xxxxxxxxxxxxxx>
  - *Date:* Wed, 28 Feb 2007 14:41:57 +0200
- 

Hello,

This patch splits the `vlan_group` struct into a multi-allocated struct. On `x86_64`, the size of the original struct is a little more than 32KB, causing a 4-order allocation, which is prone to problems caused by buddy-system external fragmentation conditions.

I couldn't just use `vmalloc()` because `vfree()` cannot be called in the `softirq` context of the RCU callback.

Signed-off-by: Dan Aloni <da-x@xxxxxxxxxxxxxx>

-----  
commit 3e6b63d0827461154066ff4c51f295bfd5006bd7  
tree ccd06cfd2acd6e8f2d07dd2fa2f5cf31dc0c1011  
parent 8d1117a9f5d302d8d460f7ef322b382e45c9ce  
author Dan Aloni <da-x@xxxxxxxxxxxxxx> Wed, 28 Feb 2007 14:20:36 +0200  
committer Dan Aloni <da-x@xxxxxxxxxxxxxx> Wed, 28 Feb 2007 14:20:36 +0200

```
drivers/net/8139cp.c | 3 +--  
drivers/net/acenic.c | 5 +-----  
drivers/net/amd8111e.c | 3 +--  
drivers/net/bnx2.c | 4 +----  
drivers/net/bonding/bond_main.c | 14 ++++++-----  
drivers/net/chelsio/cxgb2.c | 3 +--  
drivers/net/e1000/e1000_main.c | 13 +++++-----  
drivers/net/eha/eha_main.c | 3 +--  
drivers/net/gianfar.c | 3 +--  
drivers/net/ixgb/ixgb_main.c | 5 ++----  
drivers/net/ns83820.c | 3 +--  
drivers/net/r8169.c | 3 +--  
drivers/net/s2io.c | 3 +--  
drivers/net/sky2.c | 3 +--  
drivers/net/starfire.c | 5 ++----  
drivers/net/tg3.c | 3 +--  
drivers/net/typhoon.c | 3 +--  
drivers/s390/net/qeth_main.c | 25 ++++++-----  
include/linux/if_vlan.h | 25 ++++++-----
```

[PATCH] vlan & net drivers: avoid a 4-order allocation

net/8021q/vlan.c | 42 ++++++-----  
20 files changed, 96 insertions(+), 75 deletions(-)

```
diff --git a/drivers/net/8139cp.c b/drivers/net/8139cp.c
index 6f93a76..12c8453 100644
--- a/drivers/net/8139cp.c
+++ b/drivers/net/8139cp.c
@@ -448,8 +448,7 @@ static void cp_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
spin_lock_irqsave(&cp->lock, flags);
cp->cpcmd &= ~RxVlanOn;
cpw16(CpCmd, cp->cpcmd);
- if (cp->vlgrp)
- cp->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(cp->vlgrp, vid, NULL);
spin_unlock_irqrestore(&cp->lock, flags);
}
#endif /* CP_VLAN_TAG_USED */
diff --git a/drivers/net/acenic.c b/drivers/net/acenic.c
index 33c6645..7138e0e 100644
--- a/drivers/net/acenic.c
+++ b/drivers/net/acenic.c
@@ -2293,10 +2293,7 @@ static void ace_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)

local_irq_save(flags);
ace_mask_irq(dev);
-
- if (ap->vlgrp)
- ap->vlgrp->vlan_devices[vid] = NULL;
-
+ vlan_group_set_device(ap->vlgrp, vid, NULL);
ace_unmask_irq(dev);
local_irq_restore(flags);
}
diff --git a/drivers/net/amd8111e.c b/drivers/net/amd8111e.c
index 18896f2..8d57f5a 100644
--- a/drivers/net/amd8111e.c
+++ b/drivers/net/amd8111e.c
@@ -1738,8 +1738,7 @@ static void amd8111e_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
{
struct amd8111e_priv *lp = netdev_priv(dev);
spin_lock_irq(&lp->lock);
- if (lp->vlgrp)
- lp->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(lp->vlgrp, vid, NULL);
spin_unlock_irq(&lp->lock);
}
#endif
diff --git a/drivers/net/bnx2.c b/drivers/net/bnx2.c
index ee7b75b..7f5f772 100644
--- a/drivers/net/bnx2.c
+++ b/drivers/net/bnx2.c
```

## [PATCH] vlan & net drivers: avoid a 4-order allocation

```
@@ -4470,9 +4470,7 @@ bnx2_vlan_rx_kill_vid(struct net_device *dev, uint16_t vid)
struct bnx2 *bp = netdev_priv(dev);

bnx2_netif_stop(bp);
-
- if (bp->vlgrp)
- bp->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(bp->vlgrp, vid, NULL);
bnx2_set_rx_mode(dev);

bnx2_netif_start(bp);
diff --git a/drivers/net/bonding/bond_main.c b/drivers/net/bonding/bond_main.c
index 6482aed..55d3553 100644
--- a/drivers/net/bonding/bond_main.c
+++ b/drivers/net/bonding/bond_main.c
@@ -489,9 +489,9 @@ static void bond_vlan_rx_kill_vid(struct net_device *bond_dev, uint16_t vid)
/* Save and then restore vlan_dev in the grp array,
 * since the slave's driver might clear it.
 */
- vlan_dev = bond->vlgrp->vlan_devices[vid];
+ vlan_dev = vlan_group_get_device(bond->vlgrp, vid);
slave_dev->vlan_rx_kill_vid(slave_dev, vid);
- bond->vlgrp->vlan_devices[vid] = vlan_dev;
+ vlan_group_set_device(bond->vlgrp, vid, vlan_dev);
}
}

@@ -551,9 +551,9 @@ static void bond_del_vlans_from_slave(struct bonding *bond, struct net_device *s
/* Save and then restore vlan_dev in the grp array,
 * since the slave's driver might clear it.
 */
- vlan_dev = bond->vlgrp->vlan_devices[vlan->vlan_id];
+ vlan_dev = vlan_group_get_device(bond->vlgrp, vlan->vlan_id);
slave_dev->vlan_rx_kill_vid(slave_dev, vlan->vlan_id);
- bond->vlgrp->vlan_devices[vlan->vlan_id] = vlan_dev;
+ vlan_group_set_device(bond->vlgrp, vlan->vlan_id, vlan_dev);
}

unreg:
@@ -2400,7 +2400,7 @@ static void bond_arp_send_all(struct bonding *bond, struct slave *slave)
vlan_id = 0;
list_for_each_entry_safe(vlan, vlan_next, &bond->vlan_list,
vlan_list) {
- vlan_dev = bond->vlgrp->vlan_devices[vlan->vlan_id];
+ vlan_dev = vlan_group_get_device(bond->vlgrp, vlan->vlan_id);
if (vlan_dev == rt->u.dst.dev) {
vlan_id = vlan->vlan_id;
dprintk("basa: vlan match on %s %d\n",
@@ -2447,7 +2447,7 @@ static void bond_send_gratuitous_arp(struct bonding *bond)
}
```

## [PATCH] vlan & net drivers: avoid a 4-order allocation

```
list_for_each_entry(vlan, &bond->vlan_list, vlan_list) {
- vlan_dev = bond->vlgrp->vlan_devices[vlan->vlan_id];
+ vlan_dev = vlan_group_get_device(bond->vlgrp, vlan->vlan_id);
if (vlan->vlan_ip) {
bond_arp_send(slave->dev, ARPOP_REPLY, vlan->vlan_ip,
vlan->vlan_ip, vlan->vlan_id);
@@ -3374,7 +3374,7 @@ static int bond_inetaddr_event(struct notifier_block *this, unsigned long event,
void *data) {
list_for_each_entry_safe(vlan, vlan_next, &bond->vlan_list,
vlan_list) {
- vlan_dev = bond->vlgrp->vlan_devices[vlan->vlan_id];
+ vlan_dev = vlan_group_get_device(bond->vlgrp, vlan->vlan_id);
if (vlan_dev == event_dev) {
switch (event) {
case NETDEV_UP:
diff --git a/drivers/net/chelsio/cxgb2.c b/drivers/net/chelsio/cxgb2.c
index fd5d821..bdf62a4 100644
--- a/drivers/net/chelsio/cxgb2.c
+++ b/drivers/net/chelsio/cxgb2.c
@@ -919,8 +919,7 @@ static void vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
struct adapter *adapter = dev->priv;

spin_lock_irq(&adapter->async_lock);
- if (adapter->vlan_grp)
- adapter->vlan_grp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(adapter->vlan_grp, vid, NULL);
spin_unlock_irq(&adapter->async_lock);
}
#endif
diff --git a/drivers/net/e1000/e1000_main.c b/drivers/net/e1000/e1000_main.c
index c6259c7..8dbcf38 100644
--- a/drivers/net/e1000/e1000_main.c
+++ b/drivers/net/e1000/e1000_main.c
@@ -376,7 +376,7 @@ @ e1000_update_mng_vlan(struct e1000_adapter *adapter)
uint16_t vid = adapter->hw.mng_cookie.vlan_id;
uint16_t old_vid = adapter->mng_vlan_id;
if (adapter->vlgrp) {
- if (!adapter->vlgrp->vlan_devices[vid]) {
+ if (!vlan_group_get_device(adapter->vlgrp, vid)) {
if (adapter->hw.mng_cookie.status &
E1000_MNG_DHCP_COOKIE_STATUS_VLAN_SUPPORT) {
e1000_vlan_rx_add_vid(netdev, vid);
@@ -386,7 +386,7 @@ @ e1000_update_mng_vlan(struct e1000_adapter *adapter)

if ((old_vid != (uint16_t)E1000_MNG_VLAN_NONE) &&
(vid != old_vid) &&
- !adapter->vlgrp->vlan_devices[old_vid])
+ !vlan_group_get_device(adapter->vlgrp, old_vid))
e1000_vlan_rx_kill_vid(netdev, old_vid);
} else
adapter->mng_vlan_id = vid;
```

[PATCH] vlan & net drivers: avoid a 4-order allocation

[PATCH] vlan & net drivers: avoid a 4-order allocation

```
@@ -1486,7 +1486,7 @@ e1000_close(struct net_device *netdev)
if ((adapter->hw.mng_cookie.status &
E1000_MNG_DHCP_COOKIE_STATUS_VLAN_SUPPORT) &&
!(adapter->vlgrp &&
- adapter->vlgrp->vlan_devices[adapter->mng_vlan_id])) {
+ vlan_group_get_device(adapter->vlgrp, adapter->mng_vlan_id))) {
e1000_vlan_rx_kill_vid(netdev, adapter->mng_vlan_id);
}

@@ -5032,10 +5032,7 @@ e1000_vlan_rx_kill_vid(struct net_device *netdev, uint16_t vid)
uint32_t vfta, index;

e1000_irq_disable(adapter);
-
- if (adapter->vlgrp)
- adapter->vlgrp->vlan_devices[vid] = NULL;
-
+ vlan_group_set_device(adapter->vlgrp, vid, NULL);
e1000_irq_enable(adapter);

if ((adapter->hw.mng_cookie.status &
@@ -5061,7 +5058,7 @@ e1000_restore_vlan(struct e1000_adapter *adapter)
if (adapter->vlgrp) {
uint16_t vid;
for (vid = 0; vid < VLAN_GROUP_ARRAY_LEN; vid++) {
- if (!adapter->vlgrp->vlan_devices[vid])
+ if (!vlan_group_get_device(adapter->vlgrp, vid))
continue;
e1000_vlan_rx_add_vid(adapter->netdev, vid);
}
diff --git a/drivers/net/ehea/ehea_main.c b/drivers/net/ehea/ehea_main.c
index 9de2d38..82d4139 100644
--- a/drivers/net/ehea/ehea_main.c
+++ b/drivers/net/ehea/ehea_main.c
@@ -1933,8 +1933,7 @@ static void ehea_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
int index;
u64 hret;

- if (port->vgrp)
- port->vgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(port->vgrp, vid, NULL);

cb1 = kzalloc(PAGE_SIZE, GFP_KERNEL);
if (!cb1) {
diff --git a/drivers/net/gianfar.c b/drivers/net/gianfar.c
index baa3514..5acebe6 100644
--- a/drivers/net/gianfar.c
+++ b/drivers/net/gianfar.c
@@ -1132,8 +1132,7 @@ static void gfar_vlan_rx_kill_vid(struct net_device *dev, uint16_t vid)

spin_lock_irqsave(&priv->rxlock, flags);
```

## [PATCH] vlan & net drivers: avoid a 4-order allocation

```
- if (priv->vlgrp)
- priv->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(priv->vgrp, vid, NULL);

spin_unlock_irqrestore(&priv->rxlock, flags);
}
diff --git a/drivers/net/ixgb/ixgb_main.c b/drivers/net/ixgb/ixgb_main.c
index a083a91..62fe003 100644
--- a/drivers/net/ixgb/ixgb_main.c
+++ b/drivers/net/ixgb/ixgb_main.c
@@ -2217,8 +2217,7 @@ ixgb_vlan_rx_kill_vid(struct net_device *netdev, uint16_t vid)

ixgb_irq_disable(adapter);

- if(adapter->vlgrp)
- adapter->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(adapter->vlgrp, vid, NULL);

ixgb_irq_enable(adapter);

@@ -2238,7 +2237,7 @@ ixgb_restore_vlan(struct ixgb_adapter *adapter)
if(adapter->vlgrp) {
uint16_t vid;
for(vid = 0; vid < VLAN_GROUP_ARRAY_LEN; vid++) {
- if(!adapter->vlgrp->vlan_devices[vid])
+ if(!vlan_group_get_device(adapter->vlgrp, vid))
continue;
ixgb_vlan_rx_add_vid(adapter->netdev, vid);
}
diff --git a/drivers/net/ns83820.c b/drivers/net/ns83820.c
index 568daeb..9ec6e9e 100644
--- a/drivers/net/ns83820.c
+++ b/drivers/net/ns83820.c
@@ -514,8 +514,7 @@ static void ns83820_vlan_rx_kill_vid(struct net_device *ndev, unsigned short vid)

spin_lock_irq(&dev->misc_lock);
spin_lock(&dev->tx_lock);
- if (dev->vlgrp)
- dev->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(dev->vlgrp, vid, NULL);
spin_unlock(&dev->tx_lock);
spin_unlock_irq(&dev->misc_lock);
}
diff --git a/drivers/net/r8169.c b/drivers/net/r8169.c
index 577babd..b2c62be 100644
--- a/drivers/net/r8169.c
+++ b/drivers/net/r8169.c
@@ -890,8 +890,7 @@ static void rtl8169_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
unsigned long flags;
```

[PATCH] vlan & net drivers: avoid a 4-order allocation

```
spin_lock_irqsave(&tp->lock, flags);
- if (tp->vlgrp)
- tp->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(tp->vlgrp, vid, NULL);
spin_unlock_irqrestore(&tp->lock, flags);
}
```

diff --git a/drivers/net/s2io.c b/drivers/net/s2io.c

index 1dd66b8..f5973e7 100644

--- a/drivers/net/s2io.c

+++ b/drivers/net/s2io.c

```
@@ -301,8 +301,7 @@ static void s2io_vlan_rx_kill_vid(struct net_device *dev, unsigned long vid)
unsigned long flags;
```

```
spin_lock_irqsave(&nic->tx_lock, flags);
- if (nic->vlgrp)
- nic->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(nic->vlgrp, vid, NULL);
spin_unlock_irqrestore(&nic->tx_lock, flags);
}
```

diff --git a/drivers/net/sky2.c b/drivers/net/sky2.c

index 822dd0b..51044cd 100644

--- a/drivers/net/sky2.c

+++ b/drivers/net/sky2.c

```
@@ -1000,8 +1000,7 @@ static void sky2_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
```

```
sky2_write32(hw, SK_REG(port, RX_GMF_CTRL_T), RX_VLAN_STRIP_OFF);
```

```
sky2_write32(hw, SK_REG(port, TX_GMF_CTRL_T), TX_VLAN_TAG_OFF);
```

```
- if (sky2->vlgrp)
```

```
- sky2->vlgrp->vlan_devices[vid] = NULL;
```

```
+ vlan_group_set_device(sky2->vlgrp, vid, NULL);
```

```
netif_tx_unlock_bh(dev);
```

```
}
```

diff --git a/drivers/net/starfire.c b/drivers/net/starfire.c

index bf873ea..8bba2e3 100644

--- a/drivers/net/starfire.c

+++ b/drivers/net/starfire.c

```
@@ -677,8 +677,7 @@ static void netdev_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
```

```
spin_lock(&np->lock);
```

```
if (debug > 1)
```

```
printk("%s: removing vlanid %d from vlan filter\n", dev->name, vid);
```

```
- if (np->vlgrp)
```

```
- np->vlgrp->vlan_devices[vid] = NULL;
```

```
+ vlan_group_set_device(np->vlgrp, vid, NULL);
```

```
set_rx_mode(dev);
```

```
spin_unlock(&np->lock);
```

```
}
```

```
@@ -1738,7 +1737,7 @@ static void set_rx_mode(struct net_device *dev)
```

```
int vlan_count = 0;
```

[PATCH] vlan & net drivers: avoid a 4-order allocation

[PATCH] vlan & net drivers: avoid a 4-order allocation

```
void __iomem *filter_addr = ioaddr + HashTable + 8;
for (i = 0; i < VLAN_VID_MASK; i++) {
- if (np->vlgrp->vlan_devices[i]) {
+ if (vlan_group_get_device(np->vlgrp, i)) {
if (vlan_count >= 32)
break;
writew(cpu_to_be16(i), filter_addr);
diff --git a/drivers/net/tg3.c b/drivers/net/tg3.c
index f4bf62c..6dcab8f 100644
--- a/drivers/net/tg3.c
+++ b/drivers/net/tg3.c
@@ -9116,8 +9116,7 @@ static void tg3_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
tg3_netif_stop(tp);
```

```
tg3_full_lock(tp, 0);
- if (tp->vlgrp)
- tp->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(tp->vlgrp, vid, NULL);
tg3_full_unlock(tp);
```

```
if (netif_running(dev))
diff --git a/drivers/net/typhoon.c b/drivers/net/typhoon.c
index 9781b16..0d91d09 100644
--- a/drivers/net/typhoon.c
+++ b/drivers/net/typhoon.c
@@ -746,8 +746,7 @@ typhoon_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
{
struct typhoon *tp = netdev_priv(dev);
spin_lock_bh(&tp->state_lock);
- if (tp->vlgrp)
- tp->vlgrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(tp->vlgrp, vid, NULL);
spin_unlock_bh(&tp->state_lock);
}
```

```
diff --git a/drivers/s390/net/qeth_main.c b/drivers/s390/net/qeth_main.c
index d2efa5f..0a7654e 100644
--- a/drivers/s390/net/qeth_main.c
+++ b/drivers/s390/net/qeth_main.c
@@ -3654,7 +3654,7 @@ qeth_verify_vlan_dev(struct net_device *dev, struct qeth_card *card)
return rc;
```

```
for (i = 0; i < VLAN_GROUP_ARRAY_LEN; i++){
- if (vg->vlan_devices[i] == dev){
+ if (vlan_group_get_device(vg, i) == dev){
rc = QETH_VLAN_CARD;
break;
}
@@ -5261,7 +5261,7 @@ qeth_free_vlan_addresses4(struct qeth_card *card, unsigned short vid)
QETH_DBF_TEXT(trace, 4, "frvaddr4");
```

## [PATCH] vlan & net drivers: avoid a 4-order allocation

```
rcu_read_lock();
- in_dev = __in_dev_get_rcu(card->vlangrp->vlan_devices[vid]);
+ in_dev = __in_dev_get_rcu(vlan_group_get_device(card->vlangrp, vid));
if (!in_dev)
goto out;
for (ifa = in_dev->ifa_list; ifa; ifa = ifa->ifa_next) {
@@ -5288,7 +5288,7 @@ qeth_free_vlan_addresses6(struct qeth_card *card, unsigned short vid)

QETH_DBF_TEXT(trace, 4, "frvaddr6");

- in6_dev = in6_dev_get(card->vlangrp->vlan_devices[vid]);
+ in6_dev = in6_dev_get(vlan_group_get_device(card->vlangrp, vid));
if (!in6_dev)
return;
for (ifa = in6_dev->addr_list; ifa; ifa = ifa->lst_next){
@@ -5360,7 +5360,7 @@ qeth_layer2_process_vlans(struct qeth_card *card, int clear)
if (!card->vlangrp)
return;
for (i = 0; i < VLAN_GROUP_ARRAY_LEN; i++) {
- if (card->vlangrp->vlan_devices[i] == NULL)
+ if (vlan_group_get_device(card->vlangrp, i) == NULL)
continue;
if (clear)
qeth_layer2_send_setdelvlan(card, i, IPA_CMD_DELVLAN);
@@ -5398,8 +5398,7 @@ qeth_vlan_rx_kill_vid(struct net_device *dev, unsigned short vid)
spin_lock_irqsave(&card->vlanlock, flags);
/* unregister IP addresses of vlan device */
qeth_free_vlan_addresses(card, vid);
- if (card->vlangrp)
- card->vlangrp->vlan_devices[vid] = NULL;
+ vlan_group_set_device(card->vlangrp, vid, NULL);
spin_unlock_irqrestore(&card->vlanlock, flags);
if (card->options.layer2)
qeth_layer2_send_setdelvlan(card, vid, IPA_CMD_DELVLAN);
@@ -5662,10 +5661,11 @@ qeth_add_vlan_mc(struct qeth_card *card)

vg = card->vlangrp;
for (i = 0; i < VLAN_GROUP_ARRAY_LEN; i++) {
- if (vg->vlan_devices[i] == NULL ||
- !(vg->vlan_devices[i]->flags & IFF_UP))
+ struct net_device *netdev = vlan_group_get_device(vg, i);
+ if (netdev == NULL ||
+ !(netdev->flags & IFF_UP))
continue;
- in_dev = in_dev_get(vg->vlan_devices[i]);
+ in_dev = in_dev_get(netdev);
if (!in_dev)
continue;
read_lock(&in_dev->mc_list_lock);
@@ -5749,10 +5749,11 @@ qeth_add_vlan_mc6(struct qeth_card *card)
```

## [PATCH] vlan & net drivers: avoid a 4-order allocation

```
vg = card->vlangrp;
for (i = 0; i < VLAN_GROUP_ARRAY_LEN; i++) {
- if (vg->vlan_devices[i] == NULL ||
- !(vg->vlan_devices[i]->flags & IFF_UP))
+ struct net_device *netdev = vlan_group_get_device(vg, i);
+ if (netdev == NULL ||
+ !(netdev->flags & IFF_UP))
continue;
- in_dev = in6_dev_get(vg->vlan_devices[i]);
+ in_dev = in6_dev_get(netdev);
if (!in_dev)
continue;
read_lock(&in_dev->lock);
diff --git a/include/linux/if_vlan.h b/include/linux/if_vlan.h
index 35cb385..d103580 100644
--- a/include/linux/if_vlan.h
+++ b/include/linux/if_vlan.h
@@ -70,15 +70,34 @@ extern void vlan_ioctl_set(int (*hook)(void __user *));
* depends on completely exhausting the VLAN identifier space. Thus
* it gives constant time look-up, but in many cases it wastes memory.
*/
-#define VLAN_GROUP_ARRAY_LEN 4096
+#define VLAN_GROUP_ARRAY_LEN 4096
+#define VLAN_GROUP_ARRAY_SPLIT_PARTS 8
+#define VLAN_GROUP_ARRAY_PART_LEN
(VLAN_GROUP_ARRAY_LEN/VLAN_GROUP_ARRAY_SPLIT_PARTS)

struct vlan_group {
int real_dev_ifindex; /* The ifindex of the ethernet(like) device the vlan is attached to. */
struct hlist_node hlist; /* linked list */
- struct net_device *vlan_devices[VLAN_GROUP_ARRAY_LEN];
+ struct net_device **vlan_devices_arrays[VLAN_GROUP_ARRAY_SPLIT_PARTS];
struct rcu_head rcu;
};

+static inline struct net_device *vlan_group_get_device(struct vlan_group *vg, int vlan_id)
+{
+ struct net_device **array;
+ array = vg->vlan_devices_arrays[vlan_id / VLAN_GROUP_ARRAY_PART_LEN];
+ return array[vlan_id % VLAN_GROUP_ARRAY_PART_LEN];
+}
+
+static inline void vlan_group_set_device(struct vlan_group *vg, int vlan_id,
+ struct net_device *dev)
+{
+ struct net_device **array;
+ if (!vg)
+ return;
+ array = vg->vlan_devices_arrays[vlan_id / VLAN_GROUP_ARRAY_PART_LEN];
+ array[vlan_id % VLAN_GROUP_ARRAY_PART_LEN] = dev;
+}

```

## [PATCH] vlan & net drivers: avoid a 4-order allocation

```
+
struct vlan_priority_tci_mapping {
unsigned long priority;
unsigned short vlan_qos; /* This should be shifted when first set, so we only do it
@@ -160,7 +179,7 @@ static inline int __vlan_hwaccel_rx(struct sk_buff *skb,
return NET_RX_DROP;
}

- skb->dev = grp->vlan_devices[vlan_tag & VLAN_VID_MASK];
+ skb->dev = vlan_group_get_device(grp, vlan_tag & VLAN_VID_MASK);
if (skb->dev == NULL) {
dev_kfree_skb_any(skb);

diff --git a/net/8021q/vlan.c b/net/8021q/vlan.c
index 18fcb9f..4378e86 100644
--- a/net/8021q/vlan.c
+++ b/net/8021q/vlan.c
@@ -184,14 +184,23 @@ struct net_device *__find_vlan_dev(struct net_device *real_dev,
struct vlan_group *grp = __vlan_find_group(real_dev->ifindex);

if (grp)
- return grp->vlan_devices[VID];
+ return vlan_group_get_device(grp, VID);

return NULL;
}

+static void vlan_group_free(struct vlan_group *grp)
+{
+ int i;
+
+ for (i=0; i < VLAN_GROUP_ARRAY_SPLIT_PARTS; i++)
+ kfree(grp->vlan_devices_arrays[i]);
+ kfree(grp);
+}
+
static void vlan_rcu_free(struct rcu_head *rcu)
{
- kfree(container_of(rcu, struct vlan_group, rcu));
+ vlan_group_free(container_of(rcu, struct vlan_group, rcu));
}

@@ -223,7 +232,7 @@ static int unregister_vlan_dev(struct net_device *real_dev,
ret = 0;

if (grp) {
- dev = grp->vlan_devices[vlan_id];
+ dev = vlan_group_get_device(grp, vlan_id);
if (dev) {
/* Remove proc entry */
```

## [PATCH] vlan & net drivers: avoid a 4-order allocation

```
vlan_proc_rem_dev(dev);
@@ -237,7 +246,7 @@ static int unregister_vlan_dev(struct net_device *real_dev,
real_dev->vlan_rx_kill_vid(real_dev, vlan_id);
}

- grp->vlan_devices[vlan_id] = NULL;
+ vlan_group_set_device(grp, vlan_id, NULL);
synchronize_net();

@@ -251,7 +260,7 @@ static int unregister_vlan_dev(struct net_device *real_dev,
* group.
*/
for (i = 0; i < VLAN_VID_MASK; i++)
- if (grp->vlan_devices[i])
+ if (vlan_group_get_device(grp, i))
break;

if (i == VLAN_VID_MASK) {
@@ -379,6 +388,7 @@ static struct net_device *register_vlan_device(const char *eth_IF_name,
struct net_device *new_dev;
struct net_device *real_dev; /* the ethernet device */
char name[IFNAMSIZ];
+ int i;

#ifdef VLAN_DEBUG
printk(VLAN_DBG "%s: if_name -:%s:- vid: %i\n",
@@ -543,6 +553,15 @@ static struct net_device *register_vlan_device(const char *eth_IF_name,
grp = kzalloc(sizeof(struct vlan_group), GFP_KERNEL);
if (!grp)
goto out_free_unregister;
+
+ for (i=0; i < VLAN_GROUP_ARRAY_SPLIT_PARTS; i++) {
+ grp->vlan_devices_arrays[i] = kzalloc(
+ sizeof(struct net_device *)*VLAN_GROUP_ARRAY_PART_LEN,
+ GFP_KERNEL);
+
+ if (!grp->vlan_devices_arrays[i])
+ goto out_free_arrays;
+ }

/* printk(KERN_ALERT "VLAN REGISTER: Allocated new group.\n"); */
grp->real_dev_ifindex = real_dev->ifindex;
@@ -554,7 +573,7 @@ static struct net_device *register_vlan_device(const char *eth_IF_name,
real_dev->vlan_rx_register(real_dev, grp);
}

- grp->vlan_devices[VLAN_ID] = new_dev;
+ vlan_group_set_device(grp, VLAN_ID, new_dev);

if (vlan_proc_add_dev(new_dev)<0)/* create it's proc entry */
```

## [PATCH] vlan & net drivers: avoid a 4-order allocation

```
printk(KERN_WARNING "VLAN: failed to add proc entry for %s\n",
@@ -571,6 +590,9 @@ static struct net_device *register_vlan_device(const char *eth_IF_name,
#endif
return new_dev;

+out_free_arrays:
+ vlan_group_free(grp);
+
out_free_unregister:
unregister_netdev(new_dev);
goto out_unlock;
@@ -606,7 +628,7 @@ static int vlan_device_event(struct notifier_block *unused, unsigned long event,
case NETDEV_CHANGE:
/* Propagate real device state to vlan devices */
for (i = 0; i < VLAN_GROUP_ARRAY_LEN; i++) {
- vlandev = grp->vlan_devices[i];
+ vlandev = vlan_group_get_device(grp, i);
if (!vlandev)
continue;

@@ -617,7 +639,7 @@ static int vlan_device_event(struct notifier_block *unused, unsigned long event,
case NETDEV_DOWN:
/* Put all VLANs for this dev in the down state too. */
for (i = 0; i < VLAN_GROUP_ARRAY_LEN; i++) {
- vlandev = grp->vlan_devices[i];
+ vlandev = vlan_group_get_device(grp, i);
if (!vlandev)
continue;

@@ -632,7 +654,7 @@ static int vlan_device_event(struct notifier_block *unused, unsigned long event,
case NETDEV_UP:
/* Put all VLANs for this dev in the up state too. */
for (i = 0; i < VLAN_GROUP_ARRAY_LEN; i++) {
- vlandev = grp->vlan_devices[i];
+ vlandev = vlan_group_get_device(grp, i);
if (!vlandev)
continue;

@@ -649,7 +671,7 @@ static int vlan_device_event(struct notifier_block *unused, unsigned long event,
for (i = 0; i < VLAN_GROUP_ARRAY_LEN; i++) {
int ret;

- vlandev = grp->vlan_devices[i];
+ vlandev = vlan_group_get_device(grp, i);
if (!vlandev)
continue;

--
```

Dan Aloni

[PATCH] vlan & net drivers: avoid a 4-order allocation

XIV LTD, <http://www.xivstorage.com>

da-x (at) monatomic.org, dan (at) xiv.co.il

–

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>