Subject: [PATCH 8/9] Small cleanup for xxx_put after evictor consolidation
Posted by Pavel Emelianov on Fri, 12 Oct 2007 13:27:40 GMT
View Forum Message <> Reply to Message

After the evictor code is consolidated there is no need in
passing the extra pointer to the xxx_put() functions.

The only place when it made sense was the evictor code itself.

Maybe this change must got with the previous (or with the
next) patch, but I try to make them shorter as much as
possible to simplify the review (but they are still large
anyway), so this change goes in a separate patch.

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

```
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index a59ac39..4ea7948 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -152,10 +152,10 @@ static __inline__ struct ipq *frag_alloc_queue(void)

 /* Destruction primitives. */

-static __inline__ void ipq_put(struct ipq *ipq, int *work)
+static __inline__ void ipq_put(struct ipq *ipq)
 {
  if (atomic_dec_and_test(&ipq->q.refcnt))
-  inet_frag_destroy(&ipq->q, &ip4_frags, work);
+  inet_frag_destroy(&ipq->q, &ip4_frags, NULL);
 }

 /* Kill ipq entry. It is not destroyed immediately,
@@ -227,7 +205,7 @@ static void ip_expire(unsigned long arg)
  }
 out:
  spin_unlock(&qp->q.lock);
- ipq_put(qp, NULL);
+ ipq_put(qp);
 }

 /* Creation primitives. */
@@ -257,7 +235,7 @@ static struct ipq *ip_frag_intern(struct ipq *qp_in)
    atomic_inc(&qp->q.refcnt);
    write_unlock(&ip4_frags.lock);
    qp_in->q.last_in |= COMPLETE;
```

```
-   ipq_put(qp_in, NULL);
+   ipq_put(qp_in);
    return qp;
   }
  }
@@ -652,7 +630,7 @@ struct sk_buff *ip_defrag(struct sk_buff *skb, u32 user)
    ret = ip_frag_reasm(qp, dev);

   spin_unlock(&qp->q.lock);
-  ipq_put(qp, NULL);
+  ipq_put(qp);
   return ret;
  }

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 785f5cd..862d089 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -147,10 +147,10 @@ static inline struct nf_ct_frag6_queue *frag_alloc_queue(void)

 /* Destruction primitives. */

-static __inline__ void fq_put(struct nf_ct_frag6_queue *fq, unsigned int *work)
+static __inline__ void fq_put(struct nf_ct_frag6_queue *fq)
 {
  if (atomic_dec_and_test(&fq->q.refcnt))
-  inet_frag_destroy(&fq->q, &nf_frags, work);
+  inet_frag_destroy(&fq->q, &nf_frags, NULL);
 }

 /* Kill fq entry. It is not destroyed immediately,
@@ -206,7 +179,7 @@ static void nf_ct_frag6_expire(unsigned long data)

 out:
  spin_unlock(&fq->q.lock);
- fq_put(fq, NULL);
+ fq_put(fq);
 }

 /* Creation primitives. */
@@ -228,7 +201,7 @@ static struct nf_ct_frag6_queue *nf_ct_frag6_intern(unsigned int hash,
    atomic_inc(&fq->q.refcnt);
    write_unlock(&nf_frags.lock);
    fq_in->q.last_in |= COMPLETE;
-   fq_put(fq_in, NULL);
+   fq_put(fq_in);
    return fq;
   }
```

```
 }
@@ -719,7 +692,7 @@ struct sk_buff *nf_ct_frag6_gather(struct sk_buff *skb)
  if (nf_ct_frag6_queue(fq, clone, fhdr, nhoff) < 0) {
   spin_unlock(&fq->q.lock);
   pr_debug("Can't insert skb to queue\n");
-  fq_put(fq, NULL);
+  fq_put(fq);
   goto ret_orig;
  }

@@ -730,7 +703,7 @@ struct sk_buff *nf_ct_frag6_gather(struct sk_buff *skb)
  }
  spin_unlock(&fq->q.lock);

- fq_put(fq, NULL);
+ fq_put(fq);
  return ret_skb;

 ret_orig:
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 74b2113..454db16 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -165,10 +165,10 @@ static inline struct frag_queue *frag_alloc_queue(void)

 /* Destruction primitives. */

-static __inline__ void fq_put(struct frag_queue *fq, int *work)
+static __inline__ void fq_put(struct frag_queue *fq)
 {
  if (atomic_dec_and_test(&fq->q.refcnt))
-  inet_frag_destroy(&fq->q, &ip6_frags, work);
+  inet_frag_destroy(&fq->q, &ip6_frags, NULL);
 }

 /* Kill fq entry. It is not destroyed immediately,
@@ -246,7 +224,7 @@ out:
  if (dev)
   dev_put(dev);
  spin_unlock(&fq->q.lock);
- fq_put(fq, NULL);
+ fq_put(fq);
 }

 /* Creation primitives. */
@@ -270,7 +248,7 @@ static struct frag_queue *ip6_frag_intern(struct frag_queue *fq_in)
    atomic_inc(&fq->q.refcnt);
    write_unlock(&ip6_frags.lock);
```

```
    fq_in->q.last_in |= COMPLETE;
-   fq_put(fq_in, NULL);
+   fq_put(fq_in);
    return fq;
   }
  }
@@ -676,7 +654,7 @@ static int ipv6_frag_rcv(struct sk_buff **skbp)
    ret = ip6_frag_reasm(fq, skbp, dev);

   spin_unlock(&fq->q.lock);
-  fq_put(fq, NULL);
+  fq_put(fq);
   return ret;
  }

--
1.5.3.4
```