Subject: [PATCH 7/9] Consolidate the xxx_evictor
Posted by Pavel Emelianov on Fri, 12 Oct 2007 13:24:27 GMT
View Forum Message <> Reply to Message

The evictors collect some statistics for ipv4 and ipv6,
so make it return the number of evicted queues and account
them all at once in the caller.

The XXX_ADD_STATS_BH() macros are just for this case,
but maybe there are places in code, that can make use of
them as well.

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

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 2dd1cd4..cf583cf 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -49,5 +49,6 @@ void inet_frags_fini(struct inet_frags *);
 void inet_frag_kill(struct inet_frag_queue *q, struct inet_frags *f);
 void inet_frag_destroy(struct inet_frag_queue *q,
     struct inet_frags *f, int *work);
+int inet_frag_evictor(struct inet_frags *f);

 #endif
diff --git a/include/net/ip.h b/include/net/ip.h
index e7b0feb..00ed4f3 100644
--- a/include/net/ip.h
+++ b/include/net/ip.h
@@ -160,6 +160,7 @@ DECLARE_SNMP_STAT(struct ipstats_mib, ip_statistics);
 #define IP_INC_STATS(field)  SNMP_INC_STATS(ip_statistics, field)
 #define IP_INC_STATS_BH(field)  SNMP_INC_STATS_BH(ip_statistics, field)
 #define IP_INC_STATS_USER(field)  SNMP_INC_STATS_USER(ip_statistics, field)
+#define IP_ADD_STATS_BH(field, val) SNMP_ADD_STATS_BH(ip_statistics, field, val)
 DECLARE_SNMP_STAT(struct linux_mib, net_statistics);
 #define NET_INC_STATS(field)  SNMP_INC_STATS(net_statistics, field)
 #define NET_INC_STATS_BH(field)  SNMP_INC_STATS_BH(net_statistics, field)
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index b29d76c..a0f1042 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -120,12 +120,21 @@ extern int sysctl_mld_max_msf;
  SNMP_INC_STATS##modifier(statname##_statistics, (field)); \
 })

+#define _DEVADD(statname, modifier, idev, field, val)   \
```

```
+({        \
+ struct inet6_dev *_idev = (idev);    \
+ if (likely(_idev != NULL))     \
+  SNMP_ADD_STATS##modifier((_idev)->stats.statname, (field), (val)); \
+ SNMP_ADD_STATS##modifier(statname##_statistics, (field), (val));\
+})
+
 /* MIBs */
 DECLARE_SNMP_STAT(struct ipstats_mib, ipv6_statistics);

 #define IP6_INC_STATS(idev,field) _DEVINC(ipv6, , idev, field)
 #define IP6_INC_STATS_BH(idev,field) _DEVINC(ipv6, _BH, idev, field)
 #define IP6_INC_STATS_USER(idev,field) _DEVINC(ipv6, _USER, idev, field)
+#define IP6_ADD_STATS_BH(idev,field,val) _DEVADD(ipv6, _BH, idev, field, val)

 DECLARE_SNMP_STAT(struct icmpv6_mib, icmpv6_statistics);
 DECLARE_SNMP_STAT(struct icmpv6msg_mib, icmpv6msg_statistics);
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 15fb2c4..484cf51 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -140,3 +140,35 @@ void inet_frag_destroy(struct inet_frag_queue *q, struct inet_frags *f,

 }
 EXPORT_SYMBOL(inet_frag_destroy);
+
+int inet_frag_evictor(struct inet_frags *f)
+{
+ struct inet_frag_queue *q;
+ int work, evicted = 0;
+
+ work = atomic_read(&f->mem) - f->ctl->low_thresh;
+ while (work > 0) {
+  read_lock(&f->lock);
+  if (list_empty(&f->lru_list)) {
+   read_unlock(&f->lock);
+   break;
+  }
+
+  q = list_first_entry(&f->lru_list,
+    struct inet_frag_queue, lru_list);
+  atomic_inc(&q->refcnt);
+  read_unlock(&f->lock);
+
+  spin_lock(&q->lock);
+  if (!(q->last_in & COMPLETE))
+   inet_frag_kill(q, f);
+  spin_unlock(&q->lock);
```

```
+
+  if (atomic_dec_and_test(&q->refcnt))
+   inet_frag_destroy(q, f, &work);
+  evicted++;
+ }
+
+ return evicted;
+}
+EXPORT_SYMBOL(inet_frag_evictor);
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
@@ -171,33 +171,11 @@ static void ipq_kill(struct ipq *ipq)
  */
 static void ip_evictor(void)
 {
- struct ipq *qp;
- struct list_head *tmp;
- int work;
-
- work = atomic_read(&ip4_frags.mem) - ip4_frags_ctl.low_thresh;
- if (work <= 0)
-  return;
-
- while (work > 0) {
-  read_lock(&ip4_frags.lock);
-  if (list_empty(&ip4_frags.lru_list)) {
-   read_unlock(&ip4_frags.lock);
-   return;
-  }
-  tmp = ip4_frags.lru_list.next;
-  qp = list_entry(tmp, struct ipq, q.lru_list);
-  atomic_inc(&qp->q.refcnt);
-  read_unlock(&ip4_frags.lock);
+ int evicted;

-  spin_lock(&qp->q.lock);
-  if (!(qp->q.last_in&COMPLETE))
-   ipq_kill(qp);
-  spin_unlock(&qp->q.lock);
-
-  ipq_put(qp, &work);
-  IP_INC_STATS_BH(IPSTATS_MIB_REASMFAILS);
- }
+ evicted = inet_frag_evictor(&ip4_frags);
+ if (evicted)
+  IP_ADD_STATS_BH(IPSTATS_MIB_REASMFAILS, evicted);
```

```
 }

 /*
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
@@ -163,34 +163,7 @@ static __inline__ void fq_kill(struct nf_ct_frag6_queue *fq)

 static void nf_ct_frag6_evictor(void)
 {
- struct nf_ct_frag6_queue *fq;
- struct list_head *tmp;
- unsigned int work;
-
- work = atomic_read(&nf_frags.mem);
- if (work <= nf_frags_ctl.low_thresh)
-  return;
-
- work -= nf_frags_ctl.low_thresh;
- while (work > 0) {
-  read_lock(&nf_frags.lock);
-  if (list_empty(&nf_frags.lru_list)) {
-   read_unlock(&nf_frags.lock);
-   return;
-  }
-  tmp = nf_frags.lru_list.next;
-  BUG_ON(tmp == NULL);
-  fq = list_entry(tmp, struct nf_ct_frag6_queue, q.lru_list);
-  atomic_inc(&fq->q.refcnt);
-  read_unlock(&nf_frags.lock);
-
-  spin_lock(&fq->q.lock);
-  if (!(fq->q.last_in&COMPLETE))
-   fq_kill(fq);
-  spin_unlock(&fq->q.lock);
-
-  fq_put(fq, &work);
- }
+ inet_frag_evictor(&nf_frags);
 }

 static void nf_ct_frag6_expire(unsigned long data)
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
@@ -181,33 +181,11 @@ static __inline__ void fq_kill(struct frag_queue *fq)
```

```
 static void ip6_evictor(struct inet6_dev *idev)
 {
- struct frag_queue *fq;
- struct list_head *tmp;
- int work;
-
- work = atomic_read(&ip6_frags.mem) - ip6_frags_ctl.low_thresh;
- if (work <= 0)
-  return;
-
- while(work > 0) {
-  read_lock(&ip6_frags.lock);
-  if (list_empty(&ip6_frags.lru_list)) {
-   read_unlock(&ip6_frags.lock);
-   return;
-  }
-  tmp = ip6_frags.lru_list.next;
-  fq = list_entry(tmp, struct frag_queue, q.lru_list);
-  atomic_inc(&fq->q.refcnt);
-  read_unlock(&ip6_frags.lock);
-
-  spin_lock(&fq->q.lock);
-  if (!(fq->q.last_in&COMPLETE))
-   fq_kill(fq);
-  spin_unlock(&fq->q.lock);
+ int evicted;

-  fq_put(fq, &work);
-  IP6_INC_STATS_BH(idev, IPSTATS_MIB_REASMFAILS);
- }
+ evicted = inet_frag_evictor(&ip6_frags);
+ if (evicted)
+  IP6_ADD_STATS_BH(idev, IPSTATS_MIB_REASMFAILS, evicted);
 }

 static void ip6_frag_expire(unsigned long data)
--
1.5.3.4
```