Subject: [PATCH v6 3/8] foundations of per-cgroup memory pressure controlling.
Posted by Glauber Costa on Mon, 10 Oct 2011 10:24:23 GMT
View Forum Message <> Reply to Message

This patch converts struct sock fields memory_pressure,
memory_allocated, sockets_allocated, and sysctl_mem (now prot_mem)
to function pointers, receiving a struct mem_cgroup parameter.

enter_memory_pressure is kept the same, since all its callers
have socket a context, and the kmem_cgroup can be derived from
the socket itself.

To keep things working, the patch convert all users of those fields
to use acessor functions.

In my benchmarks I didn't see a significant performance difference
with this patch applied compared to a baseline (around 1 % diff, thus
inside error margin).

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: David S. Miller <davem@davemloft.net>
CC: Hiroyouki Kamezawa <kamezawa.hiroyu@jp.fujitsu.com>
CC: Eric W. Biederman <ebiederm@xmission.com>

```
 crypto/af_alg.c               |    8 +++-
 include/linux/memcontrol.h    |   22 ++++++++-
 include/net/sock.h            |  114 ++++++++++++++++++++++++++++++++++++++++++--
 include/net/tcp.h             |   12 +++--
 include/net/udp.h             |    4 +-
 include/trace/events/sock.h   |   10 ++--
 mm/memcontrol.c               |   19 +++++++-
 net/core/sock.c               |   62 ++++++++++++++---------
 net/decnet/af_decnet.c        |   22 +++++++-
 net/ipv4/proc.c               |    7 ++-
 net/ipv4/tcp.c                |   28 +++++++++-
 net/ipv4/tcp_input.c          |   12 ++--
 net/ipv4/tcp_ipv4.c           |   12 ++--
 net/ipv4/tcp_output.c         |    2 +-
 net/ipv4/tcp_timer.c          |    2 +-
 net/ipv4/udp.c                |   21 ++++++--
 net/ipv6/tcp_ipv6.c           |   10 ++--
 net/ipv6/udp.c                |    4 +-
 net/sctp/socket.c             |   37 ++++++++++---
 19 files changed, 320 insertions(+), 88 deletions(-)
```

diff --git a/crypto/af_alg.c b/crypto/af_alg.c
index ac33d5f..09cdf11 100644
--- a/crypto/af_alg.c

```
+++ b/crypto/af_alg.c
@@ -29,10 +29,16 @@ struct alg_type_list {

 static atomic_long_t alg_memory_allocated;

+static long memory_allocated_alg(struct mem_cgroup *memcg, long val,
+      int *parent_status)
+{
+ return atomic_long_add_return(val, &alg_memory_allocated);
+}
+
 static struct proto alg_proto = {
   .name    = "ALG",
   .owner   = THIS_MODULE,
- .memory_allocated = &alg_memory_allocated,
+ .mem_allocated_add = memory_allocated_alg,
   .obj_size  = sizeof(struct alg_sock),
 };

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index 88aea1b..99a8ba2 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -361,6 +361,10 @@ static inline
 void mem_cgroup_count_vm_event(struct mm_struct *mm, enum vm_event_item idx)
 {
 }
+static inline struct mem_cgroup *mem_cgroup_from_task(struct task_struct *p)
+{
+ return NULL;
+}
 #endif /* CONFIG_CGROUP_MEM_CONT */

 #if !defined(CONFIG_CGROUP_MEM_RES_CTLR) || !defined(CONFIG_DEBUG_VM)
@@ -377,12 +381,28 @@ mem_cgroup_print_bad_page(struct page *page)
 #endif

 #ifdef CONFIG_INET
+enum {
+ UNDER_LIMIT,
+ OVER_LIMIT,
+};
+
 struct sock;
+struct proto;
 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
 void sock_update_memcg(struct sock *sk);
 void sock_release_memcg(struct sock *sk);
```

```
-
+void memcg_sockets_allocated_dec(struct mem_cgroup *memcg, struct proto *prot);
+void memcg_sockets_allocated_inc(struct mem_cgroup *memcg, struct proto *prot);
 #else
+/* memcontrol includes sockets.h, that includes memcontrol.h ... */
+static inline void memcg_sockets_allocated_dec(struct mem_cgroup *memcg,
+           struct proto *prot)
+{
+}
+static inline void memcg_sockets_allocated_inc(struct mem_cgroup *memcg,
+           struct proto *prot)
+{
+}
 static inline void sock_update_memcg(struct sock *sk)
 {
 }
diff --git a/include/net/sock.h b/include/net/sock.h
index afe1467..163f87b 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -54,6 +54,7 @@
 #include <linux/security.h>
 #include <linux/slab.h>
 #include <linux/uaccess.h>
+#include <linux/cgroup.h>

 #include <linux/filter.h>
 #include <linux/rculist_nulls.h>
@@ -168,6 +169,8 @@ struct sock_common {
  /* public: */
 };

+struct mem_cgroup;
+
 /**
   * struct sock - network layer representation of sockets
   * @__sk_common: shared layout with inet_timewait_sock
@@ -786,18 +789,36 @@ struct proto {
  unsigned int  inuse_idx;
 #endif

+ /*
+  * per-cgroup memory tracking:
+  *
+  * The following functions track memory consumption of network buffers
+  * by cgroup (kmem_cgroup) for the current protocol. As of the rest
+  * of the fields in this structure, not all protocols are required
+  * to implement them. Protocols that don't want to do per-cgroup
```

```
+  * memory pressure management, can just assume the root cgroup is used.
+  *
+  */
   /* Memory pressure */
   void   (*enter_memory_pressure)(struct sock *sk);
- atomic_long_t  *memory_allocated; /* Current allocated memory. */
- struct percpu_counter *sockets_allocated; /* Current number of sockets. */
   /*
-  * Pressure flag: try to collapse.
+  * Add a value in pages to the current memory allocation,
+  * and return the current value.
+  */
+ long   (*mem_allocated_add)(struct mem_cgroup *memcg,
+           long val, int *parent_status);
+ /* Pointer to the current number of sockets in this cgroup. */
+ struct percpu_counter *(*sockets_allocated)(const struct mem_cgroup *memcg);
+ /*
+  * Per cgroup pointer to the pressure flag: try to collapse.
    * Technical note: it is used by multiple contexts non atomically.
    * All the __sk_mem_schedule() is of this nature: accounting
    * is strict, actions are advisory and have some latency.
    */
- int   *memory_pressure;
- long   *sysctl_mem;
+ int   *(*memory_pressure)(const struct mem_cgroup *memcg);
+ /* Pointer to the per-cgroup version of the the sysctl_mem field */
+ long   *(*prot_mem)(const struct mem_cgroup *memcg);
+
   int   *sysctl_wmem;
   int   *sysctl_rmem;
   int   max_header;
@@ -856,6 +877,87 @@ static inline void sk_refcnt_debug_release(const struct sock *sk)
  #define sk_refcnt_debug_release(sk) do { } while (0)
  #endif /* SOCK_REFCNT_DEBUG */

+#include <linux/memcontrol.h>
+static inline int *sk_memory_pressure(struct sock *sk)
+{
+ int *ret = NULL;
+ if (sk->sk_prot->memory_pressure)
+  ret = sk->sk_prot->memory_pressure(sk->sk_cgrp);
+ return ret;
+}
+
+static inline long sk_prot_mem(struct sock *sk, int index)
+{
+ long *prot = sk->sk_prot->prot_mem(sk->sk_cgrp);
+ return prot[index];
```

```
+}
+
+static inline long
+sk_memory_allocated(struct sock *sk)
+{
+ struct proto *prot = sk->sk_prot;
+ struct mem_cgroup *cg = sk->sk_cgrp;
+
+ return prot->mem_allocated_add(cg, 0, NULL);
+}
+
+static inline long
+sk_memory_allocated_add(struct sock *sk, int amt, int *parent_status)
+{
+ struct proto *prot = sk->sk_prot;
+ struct mem_cgroup *cg = sk->sk_cgrp;
+
+ return prot->mem_allocated_add(cg, amt, parent_status);
+}
+
+static inline void
+sk_memory_allocated_sub(struct sock *sk, int amt, int parent_status)
+{
+ struct proto *prot = sk->sk_prot;
+ struct mem_cgroup *cg = sk->sk_cgrp;
+
+ prot->mem_allocated_add(cg, -amt, &parent_status);
+}
+
+static inline void sk_sockets_allocated_dec(struct sock *sk)
+{
+ struct proto *prot = sk->sk_prot;
+ struct mem_cgroup *cg = sk->sk_cgrp;
+
+ percpu_counter_dec(prot->sockets_allocated(cg));
+ memcg_sockets_allocated_dec(cg, prot);
+}
+
+static inline void sk_sockets_allocated_inc(struct sock *sk)
+{
+ struct proto *prot = sk->sk_prot;
+ struct mem_cgroup *cg = sk->sk_cgrp;
+
+ percpu_counter_inc(prot->sockets_allocated(cg));
+ memcg_sockets_allocated_inc(cg, prot);
+}
+
+static inline int
```

```
+sk_sockets_allocated_read_positive(struct sock *sk)
+{
+ struct proto *prot = sk->sk_prot;
+ struct mem_cgroup *cg = sk->sk_cgrp;
+
+ return percpu_counter_sum_positive(prot->sockets_allocated(cg));
+}
+
+static inline int
+kcg_sockets_allocated_sum_positive(struct proto *prot, struct mem_cgroup *cg)
+{
+ return percpu_counter_sum_positive(prot->sockets_allocated(cg));
+}
+
+static inline long
+kcg_memory_allocated(struct proto *prot, struct mem_cgroup *cg)
+{
+ return prot->mem_allocated_add(cg, 0, NULL);
+}
+

 #ifdef CONFIG_PROC_FS
 /* Called with local bh disabled */
@@ -952,7 +1054,7 @@ static inline int sk_mem_pages(int amt)
 static inline int sk_has_account(struct sock *sk)
 {
  /* return true if protocol supports memory accounting */
- return !!sk->sk_prot->memory_allocated;
+ return !!sk->sk_prot->mem_allocated_add;
 }

 static inline int sk_wmem_schedule(struct sock *sk, int size)
diff --git a/include/net/tcp.h b/include/net/tcp.h
index acc620a..eac7bf6 100644
--- a/include/net/tcp.h
+++ b/include/net/tcp.h
@@ -45,6 +45,7 @@
 #include <net/dst.h>

 #include <linux/seq_file.h>
+#include <linux/memcontrol.h>

 extern struct inet_hashinfo tcp_hashinfo;

@@ -253,9 +254,12 @@ extern int sysctl_tcp_cookie_size;
 extern int sysctl_tcp_thin_linear_timeouts;
 extern int sysctl_tcp_thin_dupack;
```

```
-extern atomic_long_t tcp_memory_allocated;
-extern struct percpu_counter tcp_sockets_allocated;
-extern int tcp_memory_pressure;
+struct mem_cgroup;
+extern long *tcp_sysctl_mem(const struct mem_cgroup *memcg);
+struct percpu_counter *sockets_allocated_tcp(const struct mem_cgroup *memcg);
+int *memory_pressure_tcp(const struct mem_cgroup *memcg);
+long memory_allocated_tcp_add(struct mem_cgroup *memcg, long val,
+         int *parent_status);

 /*
  * The next routines deal with comparing 32 bit unsigned ints
@@ -286,7 +290,7 @@ static inline bool tcp_too_many_orphans(struct sock *sk, int shift)
  }

  if (sk->sk_wmem_queued > SOCK_MIN_SNDBUF &&
-     atomic_long_read(&tcp_memory_allocated) > sysctl_tcp_mem[2])
+     sk_memory_allocated(sk) > sk_prot_mem(sk, 2))
   return true;
  return false;
 }
diff --git a/include/net/udp.h b/include/net/udp.h
index 67ea6fc..eecd727 100644
--- a/include/net/udp.h
+++ b/include/net/udp.h
@@ -105,7 +105,9 @@ static inline struct udp_hslot *udp_hashslot2(struct udp_table *table,

 extern struct proto udp_prot;

-extern atomic_long_t udp_memory_allocated;
+long memory_allocated_udp_add(struct mem_cgroup *memcg, long val,
+         int *parent_status);
+long *udp_sysctl_mem(const struct mem_cgroup *memcg);

 /* sysctl variables for udp */
 extern long sysctl_udp_mem[3];
diff --git a/include/trace/events/sock.h b/include/trace/events/sock.h
index 779abb9..12a6083 100644
--- a/include/trace/events/sock.h
+++ b/include/trace/events/sock.h
@@ -37,7 +37,7 @@ TRACE_EVENT(sock_exceed_buf_limit,

  TP_STRUCT__entry(
   __array(char, name, 32)
-  __field(long *, sysctl_mem)
+  __field(long *, prot_mem)
   __field(long, allocated)
   __field(int, sysctl_rmem)
```

```
   __field(int, rmem_alloc)
@@ -45,7 +45,7 @@ TRACE_EVENT(sock_exceed_buf_limit,

  TP_fast_assign(
   strncpy(__entry->name, prot->name, 32);
-  __entry->sysctl_mem = prot->sysctl_mem;
+  __entry->prot_mem = sk->sk_prot->prot_mem(sk->sk_cgrp);
   __entry->allocated = allocated;
   __entry->sysctl_rmem = prot->sysctl_rmem[0];
   __entry->rmem_alloc = atomic_read(&sk->sk_rmem_alloc);
@@ -54,9 +54,9 @@ TRACE_EVENT(sock_exceed_buf_limit,
  TP_printk("proto:%s sysctl_mem=%ld,%ld,%ld allocated=%ld "
   "sysctl_rmem=%d rmem_alloc=%d",
   __entry->name,
-  __entry->sysctl_mem[0],
-  __entry->sysctl_mem[1],
-  __entry->sysctl_mem[2],
+  __entry->prot_mem[0],
+  __entry->prot_mem[1],
+  __entry->prot_mem[2],
   __entry->allocated,
   __entry->sysctl_rmem,
   __entry->rmem_alloc)
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 303d9d8..ad55b28 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -376,6 +376,7 @@ enum mem_type {
 #define MEM_CGROUP_RECLAIM_SOFT_BIT 0x2
 #define MEM_CGROUP_RECLAIM_SOFT  (1 << MEM_CGROUP_RECLAIM_SOFT_BIT)

+static struct mem_cgroup *parent_mem_cgroup(struct mem_cgroup *memcg);
 /* Writing them here to avoid exposing memcg's inner layout */
 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
 #ifdef CONFIG_INET
@@ -409,13 +410,27 @@ void sock_release_memcg(struct sock *sk)
 {
  cgroup_release_and_wakeup_rmdir(mem_cgroup_css(sk->sk_cgrp));
 }
+
+void memcg_sockets_allocated_dec(struct mem_cgroup *memcg, struct proto *prot)
+{
+ memcg = parent_mem_cgroup(memcg);
+ for (; memcg; memcg = parent_mem_cgroup(memcg))
+  percpu_counter_dec(prot->sockets_allocated(memcg));
+}
+EXPORT_SYMBOL(memcg_sockets_allocated_dec);
+
```

```
+void memcg_sockets_allocated_inc(struct mem_cgroup *memcg, struct proto *prot)
+{
+ memcg = parent_mem_cgroup(memcg);
+ for (; memcg; memcg = parent_mem_cgroup(memcg))
+  percpu_counter_inc(prot->sockets_allocated(memcg));
+}
+EXPORT_SYMBOL(memcg_sockets_allocated_inc);
 #endif /* CONFIG_INET */
 #endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */

-
 static void mem_cgroup_get(struct mem_cgroup *mem);
 static void mem_cgroup_put(struct mem_cgroup *mem);
-static struct mem_cgroup *parent_mem_cgroup(struct mem_cgroup *mem);
 static void drain_all_stock_async(struct mem_cgroup *mem);

 static struct mem_cgroup_per_zone *
diff --git a/net/core/sock.c b/net/core/sock.c
index 5426ba0..22ef143 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -1293,7 +1293,7 @@ struct sock *sk_clone(const struct sock *sk, const gfp_t priority)
   newsk->sk_wq = NULL;

   if (newsk->sk_prot->sockets_allocated)
-   percpu_counter_inc(newsk->sk_prot->sockets_allocated);
+   sk_sockets_allocated_inc(newsk);

   if (sock_flag(newsk, SOCK_TIMESTAMP) ||
       sock_flag(newsk, SOCK_TIMESTAMPING_RX_SOFTWARE))
@@ -1684,30 +1684,33 @@ int __sk_mem_schedule(struct sock *sk, int size, int kind)
  struct proto *prot = sk->sk_prot;
  int amt = sk_mem_pages(size);
  long allocated;
+ int *memory_pressure;
+ int parent_status = UNDER_LIMIT;

  sk->sk_forward_alloc += amt * SK_MEM_QUANTUM;
- allocated = atomic_long_add_return(amt, prot->memory_allocated);
+
+ memory_pressure = sk_memory_pressure(sk);
+ allocated = sk_memory_allocated_add(sk, amt, &parent_status);
+
+ /* Over hard limit (we, or our parents) */
+ if ((parent_status == OVER_LIMIT) || (allocated > sk_prot_mem(sk, 2)))
+  goto suppress_allocation;

  /* Under limit. */
```

```
- if (allocated <= prot->sysctl_mem[0]) {
-  if (prot->memory_pressure && *prot->memory_pressure)
-   *prot->memory_pressure = 0;
-  return 1;
- }
+ if (allocated <= sk_prot_mem(sk, 0))
+  if (memory_pressure && *memory_pressure)
+   *memory_pressure = 0;

  /* Under pressure. */
- if (allocated > prot->sysctl_mem[1])
+ if (allocated > sk_prot_mem(sk, 1))
   if (prot->enter_memory_pressure)
    prot->enter_memory_pressure(sk);

- /* Over hard limit. */
- if (allocated > prot->sysctl_mem[2])
-  goto suppress_allocation;
-
  /* guarantee minimum buffer size under pressure */
  if (kind == SK_MEM_RECV) {
   if (atomic_read(&sk->sk_rmem_alloc) < prot->sysctl_rmem[0])
    return 1;
+
  } else { /* SK_MEM_SEND */
   if (sk->sk_type == SOCK_STREAM) {
    if (sk->sk_wmem_queued < prot->sysctl_wmem[0])
@@ -1717,13 +1720,13 @@ int __sk_mem_schedule(struct sock *sk, int size, int kind)
     return 1;
  }

- if (prot->memory_pressure) {
+ if (memory_pressure) {
   int alloc;

-  if (!*prot->memory_pressure)
+  if (!*memory_pressure)
    return 1;
-  alloc = percpu_counter_read_positive(prot->sockets_allocated);
-  if (prot->sysctl_mem[2] > alloc *
+  alloc = sk_sockets_allocated_read_positive(sk);
+  if (sk_prot_mem(sk, 2) > alloc *
       sk_mem_pages(sk->sk_wmem_queued +
      atomic_read(&sk->sk_rmem_alloc) +
      sk->sk_forward_alloc))
@@ -1746,7 +1749,9 @@ suppress_allocation:

  /* Alas. Undo changes. */
```

```
  sk->sk_forward_alloc -= amt * SK_MEM_QUANTUM;
- atomic_long_sub(amt, prot->memory_allocated);
+
+ sk_memory_allocated_sub(sk, amt, parent_status);
+
  return 0;
 }
 EXPORT_SYMBOL(__sk_mem_schedule);
@@ -1757,15 +1762,15 @@ EXPORT_SYMBOL(__sk_mem_schedule);
  */
 void __sk_mem_reclaim(struct sock *sk)
 {
- struct proto *prot = sk->sk_prot;
+ int *memory_pressure = sk_memory_pressure(sk);

- atomic_long_sub(sk->sk_forward_alloc >> SK_MEM_QUANTUM_SHIFT,
-     prot->memory_allocated);
+ sk_memory_allocated_sub(sk,
+     sk->sk_forward_alloc >> SK_MEM_QUANTUM_SHIFT, 0);
  sk->sk_forward_alloc &= SK_MEM_QUANTUM - 1;

- if (prot->memory_pressure && *prot->memory_pressure &&
-     (atomic_long_read(prot->memory_allocated) < prot->sysctl_mem[0]))
-  *prot->memory_pressure = 0;
+ if (memory_pressure && *memory_pressure &&
+     (sk_memory_allocated(sk) < sk_prot_mem(sk, 0)))
+  *memory_pressure = 0;
 }
 EXPORT_SYMBOL(__sk_mem_reclaim);

@@ -2484,13 +2489,20 @@ static char proto_method_implemented(const void *method)

 static void proto_seq_printf(struct seq_file *seq, struct proto *proto)
 {
+ struct mem_cgroup *cg = mem_cgroup_from_task(current);
+ int *memory_pressure = NULL;
+
+ if (proto->memory_pressure)
+  memory_pressure = proto->memory_pressure(cg);
+
  seq_printf(seq, "%-9s %4u %6d  %6ld   %-3s %6u   %-3s  %-10s "
    "%2c %2c %2c %2c %2c %2c %2c %2c %2c %2c %2c %2c %2c %2c %2c %2c %2c %2c
%2c\n",
      proto->name,
      proto->obj_size,
      sock_prot_inuse_get(seq_file_net(seq), proto),
-     proto->memory_allocated != NULL ? atomic_long_read(proto->memory_allocated) : -1L,
-     proto->memory_pressure != NULL ? *proto->memory_pressure ? "yes" : "no" : "NI",
```

```
+     proto->mem_allocated_add != NULL ?
+   kcg_memory_allocated(proto, cg) : -1L,
+     memory_pressure != NULL ? *memory_pressure ? "yes" : "no" : "NI",
      proto->max_header,
      proto->slab == NULL ? "no" : "yes",
      module_name(proto->owner),
diff --git a/net/decnet/af_decnet.c b/net/decnet/af_decnet.c
index 19acd00..724ac73 100644
--- a/net/decnet/af_decnet.c
+++ b/net/decnet/af_decnet.c
@@ -458,13 +458,29 @@ static void dn_enter_memory_pressure(struct sock *sk)
  }
 }

+static long memory_allocated_dn_add(struct mem_cgroup *memcg,
+           long val, int *parent_status)
+{
+ return atomic_long_add_return(val, &decnet_memory_allocated);
+}
+
+static int *memory_pressure_dn(const struct mem_cgroup *memcg)
+{
+ return &dn_memory_pressure;
+}
+
+static long *dn_sysctl_mem(const struct mem_cgroup *memcg)
+{
+ return sysctl_decnet_mem;
+}
+
 static struct proto dn_proto = {
  .name   = "NSP",
  .owner   = THIS_MODULE,
  .enter_memory_pressure = dn_enter_memory_pressure,
- .memory_pressure = &dn_memory_pressure,
- .memory_allocated = &decnet_memory_allocated,
- .sysctl_mem  = sysctl_decnet_mem,
+ .memory_pressure = memory_pressure_dn,
+ .mem_allocated_add = memory_allocated_dn_add,
+ .prot_mem  = dn_sysctl_mem,
  .sysctl_wmem  = sysctl_decnet_wmem,
  .sysctl_rmem  = sysctl_decnet_rmem,
  .max_header  = DN_MAX_NSP_DATA_HEADER + 64,
diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index 4bfad5d..535456d 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -52,20 +52,21 @@ static int sockstat_seq_show(struct seq_file *seq, void *v)
```

```
 {
  struct net *net = seq->private;
  int orphans, sockets;
+ struct mem_cgroup *cg = mem_cgroup_from_task(current);

  local_bh_disable();
  orphans = percpu_counter_sum_positive(&tcp_orphan_count);
- sockets = percpu_counter_sum_positive(&tcp_sockets_allocated);
+ sockets = kcg_sockets_allocated_sum_positive(&tcp_prot, cg);
  local_bh_enable();

  socket_seq_show(seq);
  seq_printf(seq, "TCP: inuse %d orphan %d tw %d alloc %d mem %ld\n",
      sock_prot_inuse_get(net, &tcp_prot), orphans,
      tcp_death_row.tw_count, sockets,
-     atomic_long_read(&tcp_memory_allocated));
+     kcg_memory_allocated(&tcp_prot, cg));
  seq_printf(seq, "UDP: inuse %d mem %ld\n",
      sock_prot_inuse_get(net, &udp_prot),
-     atomic_long_read(&udp_memory_allocated));
+     kcg_memory_allocated(&udp_prot, cg));
  seq_printf(seq, "UDPLITE: inuse %d\n",
      sock_prot_inuse_get(net, &udplite_prot));
  seq_printf(seq, "RAW: inuse %d\n",
diff --git a/net/ipv4/tcp.c b/net/ipv4/tcp.c
index 46febca..dc8f01e 100644
--- a/net/ipv4/tcp.c
+++ b/net/ipv4/tcp.c
@@ -291,13 +291,11 @@ EXPORT_SYMBOL(sysctl_tcp_rmem);
 EXPORT_SYMBOL(sysctl_tcp_wmem);

 atomic_long_t tcp_memory_allocated; /* Current allocated memory. */
-EXPORT_SYMBOL(tcp_memory_allocated);

 /*
  * Current number of TCP sockets.
  */
 struct percpu_counter tcp_sockets_allocated;
-EXPORT_SYMBOL(tcp_sockets_allocated);

 /*
  * TCP splice context
@@ -315,7 +313,18 @@ struct tcp_splice_state {
  * is strict, actions are advisory and have some latency.
  */
 int tcp_memory_pressure __read_mostly;
-EXPORT_SYMBOL(tcp_memory_pressure);
+
```

```
+int *memory_pressure_tcp(const struct mem_cgroup *memcg)
+{
+ return &tcp_memory_pressure;
+}
+EXPORT_SYMBOL(memory_pressure_tcp);
+
+struct percpu_counter *sockets_allocated_tcp(const struct mem_cgroup *memcg)
+{
+ return &tcp_sockets_allocated;
+}
+EXPORT_SYMBOL(sockets_allocated_tcp);

 void tcp_enter_memory_pressure(struct sock *sk)
 {
@@ -326,6 +335,19 @@ void tcp_enter_memory_pressure(struct sock *sk)
 }
 EXPORT_SYMBOL(tcp_enter_memory_pressure);

+long *tcp_sysctl_mem(const struct mem_cgroup *memcg)
+{
+ return sysctl_tcp_mem;
+}
+EXPORT_SYMBOL(tcp_sysctl_mem);
+
+long memory_allocated_tcp_add(struct mem_cgroup *memcg, long val,
+         int *parent_status)
+{
+ return atomic_long_add_return(val, &tcp_memory_allocated);
+}
+EXPORT_SYMBOL(memory_allocated_tcp_add);
+
 /* Convert seconds to retransmits based on initial and max timeout */
 static u8 secs_to_retrans(int seconds, int timeout, int rto_max)
 {
diff --git a/net/ipv4/tcp_input.c b/net/ipv4/tcp_input.c
index d73aab3..87520ed 100644
--- a/net/ipv4/tcp_input.c
+++ b/net/ipv4/tcp_input.c
@@ -316,7 +316,7 @@ static void tcp_grow_window(struct sock *sk, struct sk_buff *skb)
  /* Check #1 */
  if (tp->rcv_ssthresh < tp->window_clamp &&
      (int)tp->rcv_ssthresh < tcp_space(sk) &&
-     !tcp_memory_pressure) {
+     !sk_memory_pressure(sk)) {
   int incr;

   /* Check #2. Increase window, if skb with such overhead
@@ -398,8 +398,8 @@ static void tcp_clamp_window(struct sock *sk)
```

```
  if (sk->sk_rcvbuf < sysctl_tcp_rmem[2] &&
      !(sk->sk_userlocks & SOCK_RCVBUF_LOCK) &&
-     !tcp_memory_pressure &&
-     atomic_long_read(&tcp_memory_allocated) < sysctl_tcp_mem[0]) {
+     !sk_memory_pressure(sk) &&
+     sk_memory_allocated(sk) < sk_prot_mem(sk, 0)) {
    sk->sk_rcvbuf = min(atomic_read(&sk->sk_rmem_alloc),
        sysctl_tcp_rmem[2]);
  }
@@ -4804,7 +4804,7 @@ static int tcp_prune_queue(struct sock *sk)

  if (atomic_read(&sk->sk_rmem_alloc) >= sk->sk_rcvbuf)
    tcp_clamp_window(sk);
- else if (tcp_memory_pressure)
+ else if (sk_memory_pressure(sk))
    tp->rcv_ssthresh = min(tp->rcv_ssthresh, 4U * tp->advmss);

  tcp_collapse_ofo_queue(sk);
@@ -4870,11 +4870,11 @@ static int tcp_should_expand_sndbuf(struct sock *sk)
    return 0;

  /* If we are under global TCP memory pressure, do not expand.  */
- if (tcp_memory_pressure)
+ if (sk_memory_pressure(sk))
    return 0;

  /* If we are under soft global TCP memory pressure, do not expand.  */
- if (atomic_long_read(&tcp_memory_allocated) >= sysctl_tcp_mem[0])
+ if (sk_memory_allocated(sk) >= sk_prot_mem(sk, 0))
    return 0;

  /* If we filled the congestion window, do not expand.  */
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index 7963e03..7072060 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -1911,7 +1911,7 @@ static int tcp_v4_init_sock(struct sock *sk)
  sk->sk_rcvbuf = sysctl_tcp_rmem[1];

  local_bh_disable();
- percpu_counter_inc(&tcp_sockets_allocated);
+ sk_sockets_allocated_inc(sk);
  local_bh_enable();

  return 0;
@@ -1967,7 +1967,7 @@ void tcp_v4_destroy_sock(struct sock *sk)
    tp->cookie_values = NULL;
```

```
  }

- percpu_counter_dec(&tcp_sockets_allocated);
+ sk_sockets_allocated_dec(sk);
 }
 EXPORT_SYMBOL(tcp_v4_destroy_sock);

@@ -2608,11 +2608,11 @@ struct proto tcp_prot = {
  .unhash   = inet_unhash,
  .get_port  = inet_csk_get_port,
  .enter_memory_pressure = tcp_enter_memory_pressure,
- .sockets_allocated = &tcp_sockets_allocated,
+ .memory_pressure = memory_pressure_tcp,
+ .sockets_allocated = sockets_allocated_tcp,
  .orphan_count  = &tcp_orphan_count,
- .memory_allocated = &tcp_memory_allocated,
- .memory_pressure = &tcp_memory_pressure,
- .sysctl_mem  = sysctl_tcp_mem,
+ .mem_allocated_add = memory_allocated_tcp_add,
+ .prot_mem  = tcp_sysctl_mem,
  .sysctl_wmem  = sysctl_tcp_wmem,
  .sysctl_rmem  = sysctl_tcp_rmem,
  .max_header  = MAX_TCP_HEADER,
diff --git a/net/ipv4/tcp_output.c b/net/ipv4/tcp_output.c
index 882e0b0..06aeb31 100644
--- a/net/ipv4/tcp_output.c
+++ b/net/ipv4/tcp_output.c
@@ -1912,7 +1912,7 @@ u32 __tcp_select_window(struct sock *sk)
  if (free_space < (full_space >> 1)) {
   icsk->icsk_ack.quick = 0;

-  if (tcp_memory_pressure)
+  if (sk_memory_pressure(sk))
    tp->rcv_ssthresh = min(tp->rcv_ssthresh,
           4U * tp->advmss);

diff --git a/net/ipv4/tcp_timer.c b/net/ipv4/tcp_timer.c
index ecd44b0..2c67617 100644
--- a/net/ipv4/tcp_timer.c
+++ b/net/ipv4/tcp_timer.c
@@ -261,7 +261,7 @@ static void tcp_delack_timer(unsigned long data)
  }

 out:
- if (tcp_memory_pressure)
+ if (sk_memory_pressure(sk))
   sk_mem_reclaim(sk);
 out_unlock:
```

```
  bh_unlock_sock(sk);
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 1b5a193..21604b4 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
@@ -120,9 +120,6 @@ EXPORT_SYMBOL(sysctl_udp_rmem_min);
 int sysctl_udp_wmem_min __read_mostly;
 EXPORT_SYMBOL(sysctl_udp_wmem_min);

-atomic_long_t udp_memory_allocated;
-EXPORT_SYMBOL(udp_memory_allocated);
-
 #define MAX_UDP_PORTS 65536
 #define PORTS_PER_CHAIN (MAX_UDP_PORTS / UDP_HTABLE_SIZE_MIN)

@@ -1918,6 +1915,20 @@ unsigned int udp_poll(struct file *file, struct socket *sock, poll_table
*wait)
 }
 EXPORT_SYMBOL(udp_poll);

+static atomic_long_t udp_memory_allocated;
+long memory_allocated_udp_add(struct mem_cgroup *memcg, long val,
+         int *parent_status)
+{
+ return atomic_long_add_return(val, &udp_memory_allocated);
+}
+EXPORT_SYMBOL(memory_allocated_udp_add);
+
+long *udp_sysctl_mem(const struct mem_cgroup *memcg)
+{
+ return sysctl_udp_mem;
+}
+EXPORT_SYMBOL(udp_sysctl_mem);
+
 struct proto udp_prot = {
  .name      = "UDP",
  .owner     = THIS_MODULE,
@@ -1936,8 +1947,8 @@ struct proto udp_prot = {
  .unhash     = udp_lib_unhash,
  .rehash     = udp_v4_rehash,
  .get_port    = udp_v4_get_port,
- .memory_allocated  = &udp_memory_allocated,
- .sysctl_mem    = sysctl_udp_mem,
+ .mem_allocated_add = &memory_allocated_udp_add,
+ .prot_mem    = udp_sysctl_mem,
  .sysctl_wmem    = &sysctl_udp_wmem_min,
  .sysctl_rmem    = &sysctl_udp_rmem_min,
  .obj_size    = sizeof(struct udp_sock),
```

```
diff --git a/net/ipv6/tcp_ipv6.c b/net/ipv6/tcp_ipv6.c
index 7b8fc57..bdc0003 100644
--- a/net/ipv6/tcp_ipv6.c
+++ b/net/ipv6/tcp_ipv6.c
@@ -1992,7 +1992,7 @@ static int tcp_v6_init_sock(struct sock *sk)
  sk->sk_rcvbuf = sysctl_tcp_rmem[1];

  local_bh_disable();
- percpu_counter_inc(&tcp_sockets_allocated);
+ sk_sockets_allocated_inc(sk);
  local_bh_enable();

  return 0;
@@ -2201,11 +2201,11 @@ struct proto tcpv6_prot = {
  .unhash   = inet_unhash,
  .get_port  = inet_csk_get_port,
  .enter_memory_pressure = tcp_enter_memory_pressure,
- .sockets_allocated = &tcp_sockets_allocated,
- .memory_allocated = &tcp_memory_allocated,
- .memory_pressure = &tcp_memory_pressure,
+ .sockets_allocated = sockets_allocated_tcp,
+ .mem_allocated_add = memory_allocated_tcp_add,
+ .memory_pressure = memory_pressure_tcp,
  .orphan_count  = &tcp_orphan_count,
- .sysctl_mem  = sysctl_tcp_mem,
+ .prot_mem  = tcp_sysctl_mem,
  .sysctl_wmem  = sysctl_tcp_wmem,
  .sysctl_rmem  = sysctl_tcp_rmem,
  .max_header  = MAX_TCP_HEADER,
diff --git a/net/ipv6/udp.c b/net/ipv6/udp.c
index bb95e8e..0be7cbc 100644
--- a/net/ipv6/udp.c
+++ b/net/ipv6/udp.c
@@ -1465,8 +1465,8 @@ struct proto udpv6_prot = {
  .unhash     = udp_lib_unhash,
  .rehash     = udp_v6_rehash,
  .get_port    = udp_v6_get_port,
- .memory_allocated  = &udp_memory_allocated,
- .sysctl_mem    = sysctl_udp_mem,
+ .mem_allocated_add = memory_allocated_udp_add,
+ .prot_mem    = udp_sysctl_mem,
  .sysctl_wmem    = &sysctl_udp_wmem_min,
  .sysctl_rmem    = &sysctl_udp_rmem_min,
  .obj_size    = sizeof(struct udp6_sock),
diff --git a/net/sctp/socket.c b/net/sctp/socket.c
index 836aa63..8c0cde9 100644
--- a/net/sctp/socket.c
+++ b/net/sctp/socket.c
```

```
@@ -119,11 +119,32 @@ static int sctp_memory_pressure;
 static atomic_long_t sctp_memory_allocated;
 struct percpu_counter sctp_sockets_allocated;

+static long *sctp_sysctl_mem(const struct mem_cgroup *memcg)
+{
+ return sysctl_sctp_mem;
+}
+
 static void sctp_enter_memory_pressure(struct sock *sk)
 {
  sctp_memory_pressure = 1;
 }

+static int *memory_pressure_sctp(const struct mem_cgroup *memcg)
+{
+ return &sctp_memory_pressure;
+}
+
+static long memory_allocated_sctp_add(struct mem_cgroup *memcg, long val,
+          int *parent_status)
+{
+ return atomic_long_add_return(val, &sctp_memory_allocated);
+}
+
+static struct
+percpu_counter *sockets_allocated_sctp(const struct mem_cgroup *memcg)
+{
+ return &sctp_sockets_allocated;
+}

 /* Get the sndbuf space available at the time on the association.  */
 static inline int sctp_wspace(struct sctp_association *asoc)
@@ -6831,13 +6852,13 @@ struct proto sctp_prot = {
  .unhash      = sctp_unhash,
  .get_port    = sctp_get_port,
  .obj_size    =  sizeof(struct sctp_sock),
- .sysctl_mem  =  sysctl_sctp_mem,
+ .prot_mem    =  sctp_sysctl_mem,
  .sysctl_rmem =  sysctl_sctp_rmem,
  .sysctl_wmem =  sysctl_sctp_wmem,
- .memory_pressure = &sctp_memory_pressure,
+ .memory_pressure = memory_pressure_sctp,
  .enter_memory_pressure = sctp_enter_memory_pressure,
- .memory_allocated = &sctp_memory_allocated,
- .sockets_allocated = &sctp_sockets_allocated,
+ .mem_allocated_add = memory_allocated_sctp_add,
+ .sockets_allocated = sockets_allocated_sctp,
```

```
 };

 #if defined(CONFIG_IPV6) || defined(CONFIG_IPV6_MODULE)
@@ -6863,12 +6884,12 @@ struct proto sctpv6_prot = {
  .unhash  = sctp_unhash,
  .get_port = sctp_get_port,
  .obj_size = sizeof(struct sctp6_sock),
- .sysctl_mem = sysctl_sctp_mem,
+ .prot_mem = sctp_sysctl_mem,
  .sysctl_rmem = sysctl_sctp_rmem,
  .sysctl_wmem = sysctl_sctp_wmem,
- .memory_pressure = &sctp_memory_pressure,
+ .memory_pressure = memory_pressure_sctp,
  .enter_memory_pressure = sctp_enter_memory_pressure,
- .memory_allocated = &sctp_memory_allocated,
- .sockets_allocated = &sctp_sockets_allocated,
+ .mem_allocated_add = memory_allocated_sctp_add,
+ .sockets_allocated = sockets_allocated_sctp,
 };
 #endif /* defined(CONFIG_IPV6) || defined(CONFIG_IPV6_MODULE) */
--
1.7.6.4
```