Subject: Re: [PATCH v7 0/8] Request for inclusion: tcp memory buffers
Posted by Glauber Costa on Fri, 14 Oct 2011 12:56:23 GMT
View Forum Message <> Reply to Message

On 10/14/2011 12:18 AM, David Miller wrote:
> From: Glauber Costa<glommer@parallels.com>
> Date: Fri, 14 Oct 2011 00:14:40 +0400
>
>> Are you happy, or at least willing to accept, an approach that keep
>> things as they were with cgroups *compiled out*, or were you referring
>> to not in use == compiled in, but with no users?
>
> To me these are the same exact thing, because %99 of users will be running
> a kernel with every feature turned on in the Kconfig.

Ok.
Please let me know what do you think of the following patch. It is still
crude, and applies on top of what I had, for simplicity. I can of course
rework all the series for the next submission. The main idea is:

We basically don't care about accounting if all tasks are in the same,
root, cgroup. So I am using static_branch to enable this code path when
the first !root cgroup is created - a stronger statement than just
enabled/disabled by a runtime option. This should cover every single
user that is not *actively* using cgroups (I understand that most
distros will not only compile it in, but also leave it enabled...). When
this code path is disabled, we should be doing the same as before.

If you think this approach is valid, I am not left with the problem of
how to replace the fields when cgroups are enabled. But for that I guess
I can just copy the struct proto (probably only 2, or at most 3 protos
will need it anyway), and make the new sockets run on this new
structure.

Cheers

```
diff --git a/include/net/sock.h b/include/net/sock.h
index efd7664..2270e50 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -807,19 +807,35 @@ struct proto {
   * 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);
```

```
+ union {
+  long (*mem_allocated_add)(struct mem_cgroup *memcg,
+          long val, int *parent_status);
+  atomic_long_t *memory_allocated;
+ };
+
+ union {
+  /* Pointer to the current number of sockets in this cgroup. */
+  struct percpu_counter *(*sockets_allocated_cg)(const struct mem_cgroup *memcg);
+  struct percpu_counter   *sockets_allocated;
+ };
+
+ union {
   /*
    * 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)(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   *(*memory_pressure_cg)(const struct mem_cgroup *memcg);
+         int                   *memory_pressure;
+
+ };
+
+ union {
+  /* Pointer to the per-cgroup version of the the sysctl_mem field */
+  long   *(*prot_mem)(const struct mem_cgroup *memcg);
+         long                  *sysctl_mem;
+ };

   /*
    * cgroup specific init/deinit functions. Called once for all
@@ -891,85 +907,174 @@ static inline void sk_refcnt_debug_release(const struct sock *sk)
 #define sk_refcnt_debug_release(sk) do { } while (0)
 #endif /* SOCK_REFCNT_DEBUG */

+extern struct jump_label_key cgroup_crap_enabled;
 #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;
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
```

```
+ if (static_branch(&cgroup_crap_enabled)) {
+  int *ret = NULL;
+  if (!sk->sk_cgrp)
+   goto nocgroup;
+  if (sk->sk_prot->memory_pressure)
+   ret = sk->sk_prot->memory_pressure_cg(sk->sk_cgrp);
+  return ret;
+ }
+nocgroup:
+#endif
+ return sk->sk_prot->memory_pressure;
 }

 static inline long sk_prot_mem(struct sock *sk, int index)
 {
- long *prot = sk->sk_prot->prot_mem(sk->sk_cgrp);
+ long *prot;
+ prot = sk->sk_prot->prot_mem(sk->sk_cgrp);
  return prot[index];
+
+#if 0
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  long *prot;
+  if (!sk->sk_cgrp)
+   goto nocgroup;
+  prot = sk->sk_prot->prot_mem(sk->sk_cgrp);
+  return prot[index];
+ }
+nocgroup:
+#endif
+ return sk->sk_prot->sysctl_mem[index];
+#endif
 }

 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);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  struct mem_cgroup *cg = sk->sk_cgrp;
+  if (!cg) /* this handles the case with existing sockets */
+   goto nocgroup;
+  return prot->mem_allocated_add(cg, 0, NULL);
```

```
+ }
+nocgroup:
+#endif
+ return atomic_long_read(prot->memory_allocated);
 }

 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);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  struct mem_cgroup *cg = sk->sk_cgrp;
+  if (!cg)
+   goto nocgroup;
+  return prot->mem_allocated_add(cg, amt, parent_status);
+ }
+nocgroup:
+#endif
+ return atomic_long_add_return(amt, prot->memory_allocated);
 }

 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);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  struct mem_cgroup *cg = sk->sk_cgrp;
+
+  if (!cg)
+   goto nocgroup;
+
+  prot->mem_allocated_add(cg, -amt, &parent_status);
+ } else
+nocgroup:
+#endif
+ atomic_long_sub(amt, prot->memory_allocated);
 }

 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);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  struct mem_cgroup *cg = sk->sk_cgrp;
+  if (!cg)
+   goto nocgroup;
+  percpu_counter_dec(prot->sockets_allocated_cg(cg));
+  memcg_sockets_allocated_dec(cg, prot);
+ } else
+nocgroup:
+#endif
+ percpu_counter_dec(prot->sockets_allocated);
 }

 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);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  struct mem_cgroup *cg = sk->sk_cgrp;
+  if (!cg)
+   goto nocgroup;
+  percpu_counter_inc(prot->sockets_allocated_cg(cg));
+  memcg_sockets_allocated_inc(cg, prot);
+ } else
+nocgroup:
+#endif
+ percpu_counter_inc(prot->sockets_allocated);
 }

 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));
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  struct mem_cgroup *cg = sk->sk_cgrp;
```

```
+  if (!cg)
+   goto nocgroup;
+  return percpu_counter_sum_positive(prot->sockets_allocated_cg(cg));
+ } else
+nocgroup:
+#endif
+ return percpu_counter_sum_positive(prot->sockets_allocated);
 }

 static inline int
 kcg_sockets_allocated_sum_positive(struct proto *prot, struct mem_cgroup *cg)
 {
- return percpu_counter_sum_positive(prot->sockets_allocated(cg));
+
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  if (!cg)
+   goto nocgroup;
+  return percpu_counter_sum_positive(prot->sockets_allocated_cg(cg));
+ } else
+nocgroup:
+#endif
+ return percpu_counter_sum_positive(prot->sockets_allocated);
 }

 static inline long
 kcg_memory_allocated(struct proto *prot, struct mem_cgroup *cg)
 {
- return prot->mem_allocated_add(cg, 0, NULL);
+
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (static_branch(&cgroup_crap_enabled)) {
+  if (!cg)
+   goto nocgroup;
+  return prot->mem_allocated_add(cg, 0, NULL);
+ }
+nocgroup:
+#endif
+ return atomic_long_read(prot->memory_allocated);
 }


diff --git a/include/net/tcp.h b/include/net/tcp.h
index 74296f4..76301a7 100644
--- a/include/net/tcp.h
+++ b/include/net/tcp.h
@@ -275,6 +275,10 @@ int *memory_pressure_tcp(const struct mem_cgroup *memcg);
 long memory_allocated_tcp_add(struct mem_cgroup *memcg, long val,
```

```
        int *parent_status);

+extern atomic_long_t tcp_memory_allocated;
+extern int tcp_memory_pressure __read_mostly;
+extern struct percpu_counter tcp_sockets_allocated;
+
 /*
  * The next routines deal with comparing 32 bit unsigned ints
  * and worry about wraparound (automatic with unsigned arithmetic).
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index febf50a..8b67528 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -400,15 +399,22 @@ static inline bool mem_cgroup_is_root(struct mem_cgroup *memcg)

 void sock_update_memcg(struct sock *sk)
 {
+ struct mem_cgroup *memcg;
+
  /* right now a socket spends its whole life in the same cgroup */
  if (sk->sk_cgrp) {
   WARN_ON(1);
   return;
  }

- rcu_read_lock();
- sk->sk_cgrp = mem_cgroup_from_task(current);
- rcu_read_unlock();
+
+ if (static_branch(&cgroup_crap_enabled)) {
+  rcu_read_lock();
+  memcg = mem_cgroup_from_task(current);
+  if (!mem_cgroup_is_root(memcg))
+   sk->sk_cgrp = memcg;
+  rcu_read_unlock();
+ }
 }

 void sock_release_memcg(struct sock *sk)
@@ -419,7 +425,7 @@ 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));
+  percpu_counter_dec(prot->sockets_allocated_cg(memcg));
 }
 EXPORT_SYMBOL(memcg_sockets_allocated_dec);
```

```
@@ -427,7 +433,7 @@ 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));
+  percpu_counter_inc(prot->sockets_allocated_cg(memcg));
 }
 EXPORT_SYMBOL(memcg_sockets_allocated_inc);

@@ -5024,19 +5030,28 @@ static struct cftype kmem_cgroup_files[] = {
 static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
 {
  int ret = 0;
+ struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);

  ret = cgroup_add_files(cont, ss, kmem_cgroup_files,
          ARRAY_SIZE(kmem_cgroup_files));

  if (!ret)
   ret = sockets_populate(cont, ss);
+
+ if (!mem_cgroup_is_root(memcg))
+  jump_label_inc(&cgroup_crap_enabled);
+
  return ret;
 };

 static void kmem_cgroup_destroy(struct cgroup_subsys *ss,
     struct cgroup *cont)
 {
+ struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);
  sockets_destroy(cont, ss);
+
+ if (!mem_cgroup_is_root(memcg))
+  jump_label_dec(&cgroup_crap_enabled);
 }
 #else
 static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
diff --git a/net/core/sock.c b/net/core/sock.c
index 3fa3ccb..444a527 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -111,6 +111,7 @@
 #include <linux/init.h>
 #include <linux/highmem.h>
 #include <linux/user_namespace.h>
```

```
+#include <linux/jump_label.h>

 #include <asm/uaccess.h>
 #include <asm/system.h>
@@ -178,6 +179,9 @@ void sockets_destroy(struct cgroup *cgrp, struct cgroup_subsys *ss)
 static struct lock_class_key af_family_keys[AF_MAX];
 static struct lock_class_key af_family_slock_keys[AF_MAX];

+struct jump_label_key cgroup_crap_enabled;
+EXPORT_SYMBOL(cgroup_crap_enabled);
+
 /*
  * Make lock validator output more readable. (we pre-construct these
  * strings build-time, so that runtime initialization of socket
@@ -2525,8 +2529,11 @@ 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 0
+ /* not important right now */
  if (proto->memory_pressure)
   memory_pressure = proto->memory_pressure(cg);
+#endif

  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\n",
diff --git a/net/ipv4/tcp.c b/net/ipv4/tcp.c
index b1abebd..c71cfa5 100644
--- a/net/ipv4/tcp.c
+++ b/net/ipv4/tcp.c
@@ -299,8 +299,11 @@ struct tcp_splice_state {

 /* Current number of TCP sockets. */
 struct percpu_counter tcp_sockets_allocated;
+EXPORT_SYMBOL(tcp_sockets_allocated);
 atomic_long_t tcp_memory_allocated; /* Current allocated memory. */
+EXPORT_SYMBOL(tcp_memory_allocated);
 int tcp_memory_pressure __read_mostly;
+EXPORT_SYMBOL(tcp_memory_pressure);

 int *memory_pressure_tcp_nocg(const struct mem_cgroup *memcg)
 {
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index 378a31c..c9724a8 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -2608,19 +2608,15 @@ struct proto tcp_prot = {
```

```
 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
  .init_cgroup  = tcp_init_cgroup,
  .destroy_cgroup  = tcp_destroy_cgroup,
- .enter_memory_pressure = tcp_enter_memory_pressure,
- .memory_pressure = memory_pressure_tcp,
- .sockets_allocated = sockets_allocated_tcp,
- .orphan_count  = &tcp_orphan_count,
- .mem_allocated_add = memory_allocated_tcp_add,
- .prot_mem  = tcp_sysctl_mem,
+ .prot_mem  = tcp_sysctl_mem_nocg,
 #else
- .enter_memory_pressure = tcp_enter_memory_pressure_nocg,
- .memory_pressure = memory_pressure_tcp_nocg,
- .sockets_allocated = sockets_allocated_tcp_nocg,
- .mem_allocated_nocg = memory_allocated_tcp_nocg,
  .prot_mem  = tcp_sysctl_mem_nocg,
 #endif
+ .enter_memory_pressure = tcp_enter_memory_pressure,
+ .memory_pressure = &tcp_memory_pressure,
+ .sockets_allocated = &tcp_sockets_allocated,
+ .orphan_count  = &tcp_orphan_count,
+ .memory_allocated = &tcp_memory_allocated,
  .sysctl_wmem  = sysctl_tcp_wmem,
  .sysctl_rmem  = sysctl_tcp_rmem,
  .max_header  = MAX_TCP_HEADER,
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 21604b4..5fa9cf3 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
@@ -1947,7 +1947,7 @@ struct proto udp_prot = {
  .unhash     = udp_lib_unhash,
  .rehash     = udp_v4_rehash,
  .get_port    = udp_v4_get_port,
- .mem_allocated_add = &memory_allocated_udp_add,
+ .memory_allocated  = &udp_memory_allocated,
  .prot_mem    = udp_sysctl_mem,
  .sysctl_wmem    = &sysctl_udp_wmem_min,
  .sysctl_rmem    = &sysctl_udp_rmem_min,
diff --git a/net/ipv6/tcp_ipv6.c b/net/ipv6/tcp_ipv6.c
index f7f9bc2..5a976fa 100644
--- a/net/ipv6/tcp_ipv6.c
+++ b/net/ipv6/tcp_ipv6.c
@@ -2197,18 +2197,15 @@ struct proto tcpv6_prot = {
  .get_port  = inet_csk_get_port,
  .orphan_count  = &tcp_orphan_count,
 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
- .enter_memory_pressure = tcp_enter_memory_pressure,
- .sockets_allocated = sockets_allocated_tcp,
```

```
- .mem_allocated_add = memory_allocated_tcp_add,
- .memory_pressure = memory_pressure_tcp,
  .prot_mem  = tcp_sysctl_mem,
 #else
- .enter_memory_pressure = tcp_enter_memory_pressure_nocg,
- .sockets_allocated = sockets_allocated_tcp_nocg,
- .mem_allocated_add = memory_allocated_tcp_nocg,
- .memory_pressure = memory_pressure_tcp_nocg,
  .prot_mem  = tcp_sysctl_mem_nocg,
 #endif
+ .enter_memory_pressure = tcp_enter_memory_pressure_nocg,
+ .sockets_allocated = &tcp_sockets_allocated,
+ .memory_allocated = &tcp_memory_allocated,
+ .memory_pressure = &tcp_memory_pressure,
+
  .sysctl_wmem  = sysctl_tcp_wmem,
  .sysctl_rmem  = sysctl_tcp_rmem,
  .max_header  = MAX_TCP_HEADER,
```

File Attachments

1) test.patch, downloaded 674 times