Subject: [PATCH] Compact sk_stream_mem_schedule() code
Posted by Pavel Emelianov on Mon, 19 Nov 2007 12:13:44 GMT
View Forum Message <> Reply to Message

This function references sk->sk_prot->xxx for many times.
It turned out, that there's so many code in it, that gcc
cannot always optimize access to sk->sk_prot's fields.

After saving the sk->sk_prot on the stack and comparing
disassembled code, it turned out that the function became
~10 bytes shorter and made less dereferences (on i386 and
x86_64). Stack consumption didn't grow.

Besides, this patch drives most of this function into the
80 columns limit.

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

```
diff --git a/net/core/stream.c b/net/core/stream.c
index 755bacb..b2fb846 100644
--- a/net/core/stream.c
+++ b/net/core/stream.c
@@ -210,35 +210,36 @@ EXPORT_SYMBOL(__sk_stream_mem_reclaim);
 int sk_stream_mem_schedule(struct sock *sk, int size, int kind)
 {
  int amt = sk_stream_pages(size);
+ struct proto *prot = sk->sk_prot;

  sk->sk_forward_alloc += amt * SK_STREAM_MEM_QUANTUM;
- atomic_add(amt, sk->sk_prot->memory_allocated);
+ atomic_add(amt, prot->memory_allocated);

  /* Under limit. */
- if (atomic_read(sk->sk_prot->memory_allocated) < sk->sk_prot->sysctl_mem[0]) {
-  if (*sk->sk_prot->memory_pressure)
-   *sk->sk_prot->memory_pressure = 0;
+ if (atomic_read(prot->memory_allocated) < prot->sysctl_mem[0]) {
+  if (*prot->memory_pressure)
+   *prot->memory_pressure = 0;
   return 1;
  }

  /* Over hard limit. */
- if (atomic_read(sk->sk_prot->memory_allocated) > sk->sk_prot->sysctl_mem[2]) {
-  sk->sk_prot->enter_memory_pressure();
+ if (atomic_read(prot->memory_allocated) > prot->sysctl_mem[2]) {
```

```
+  prot->enter_memory_pressure();
   goto suppress_allocation;
  }

  /* Under pressure. */
- if (atomic_read(sk->sk_prot->memory_allocated) > sk->sk_prot->sysctl_mem[1])
-  sk->sk_prot->enter_memory_pressure();
+ if (atomic_read(prot->memory_allocated) > prot->sysctl_mem[1])
+  prot->enter_memory_pressure();

  if (kind) {
-  if (atomic_read(&sk->sk_rmem_alloc) < sk->sk_prot->sysctl_rmem[0])
+  if (atomic_read(&sk->sk_rmem_alloc) < prot->sysctl_rmem[0])
    return 1;
- } else if (sk->sk_wmem_queued < sk->sk_prot->sysctl_wmem[0])
+ } else if (sk->sk_wmem_queued < prot->sysctl_wmem[0])
   return 1;

- if (!*sk->sk_prot->memory_pressure ||
-     sk->sk_prot->sysctl_mem[2] > atomic_read(sk->sk_prot->sockets_allocated) *
+ if (!*prot->memory_pressure ||
+     prot->sysctl_mem[2] > atomic_read(prot->sockets_allocated) *
     sk_stream_pages(sk->sk_wmem_queued +
       atomic_read(&sk->sk_rmem_alloc) +
       sk->sk_forward_alloc))
@@ -258,7 +259,7 @@ suppress_allocation:

  /* Alas. Undo changes. */
  sk->sk_forward_alloc -= amt * SK_STREAM_MEM_QUANTUM;
- atomic_sub(amt, sk->sk_prot->memory_allocated);
+ atomic_sub(amt, prot->memory_allocated);
  return 0;
 }
```

Subject: Re: [PATCH] Compact sk_stream_mem_schedule() code
Posted by Arnaldo Carvalho de M[1] on Mon, 19 Nov 2007 19:30:59 GMT
View Forum Message <> Reply to Message

Em Mon, Nov 19, 2007 at 03:13:44PM +0300, Pavel Emelyanov escreveu:
> This function references sk->sk_prot->xxx for many times.
> It turned out, that there's so many code in it, that gcc
> cannot always optimize access to sk->sk_prot's fields.
>
> After saving the sk->sk_prot on the stack and comparing
> disassembled code, it turned out that the function became
> ~10 bytes shorter and made less dereferences (on i386 and
> x86_64). Stack consumption didn't grow.

>
> Besides, this patch drives most of this function into the
> 80 columns limit.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

I wonder if making it 'const struct proto *prot = sk->sk_prot;'

would make any difference.

Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

Subject: Re: [PATCH] Compact sk_stream_mem_schedule() code
Posted by davem on Tue, 20 Nov 2007 07:22:45 GMT
View Forum Message <> Reply to Message

From: Arnaldo Carvalho de Melo <acme@ghostprotocols.net>
Date: Mon, 19 Nov 2007 17:30:59 -0200

> Em Mon, Nov 19, 2007 at 03:13:44PM +0300, Pavel Emelyanov escreveu:
> > This function references sk->sk_prot->xxx for many times.
> > It turned out, that there's so many code in it, that gcc
> > cannot always optimize access to sk->sk_prot's fields.
> >
> > After saving the sk->sk_prot on the stack and comparing
> > disassembled code, it turned out that the function became
> > ~10 bytes shorter and made less dereferences (on i386 and
> > x86_64). Stack consumption didn't grow.
> >
> > Besides, this patch drives most of this function into the
> > 80 columns limit.
> >
> > Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> I wonder if making it 'const struct proto *prot = sk->sk_prot;'
>
> would make any difference.

Such experiments are always useful, but I doubt there will
be substantial gains in this case.

> Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

I've applied the patch, thanks Pavel.
