Subject: Re: [PATCH v7 03/10] socket: initial cgroup code.
Posted by KAMEZAWA Hiroyuki on Wed, 30 Nov 2011 01:07:38 GMT
View Forum Message <> Reply to Message

On Tue, 29 Nov 2011 21:56:54 -0200
Glauber Costa <glommer@parallels.com> wrote:

> The goal of this work is to move the memory pressure tcp
> controls to a cgroup, instead of just relying on global
> conditions.
>
> To avoid excessive overhead in the network fast paths,
> the code that accounts allocated memory to a cgroup is
> hidden inside a static_branch(). This branch is patched out
> until the first non-root cgroup is created. So when nobody
> is using cgroups, even if it is mounted, no significant performance
> penalty should be seen.
>
> This patch handles the generic part of the code, and has nothing
> tcp-specific.
>
> Signed-off-by: Glauber Costa <glommer@parallels.com>
> Acked-by: Kirill A. Shutemov<kirill@shutemov.name>
> Reviewed-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujtsu.com>
> CC: David S. Miller <davem@davemloft.net>
> CC: Eric W. Biederman <ebiederm@xmission.com>
> CC: Eric Dumazet <eric.dumazet@gmail.com>

<snip>

> +extern struct jump_label_key memcg_socket_limit_enabled;
> static inline bool sk_has_memory_pressure(const struct sock *sk)
> {
>   return sk->sk_prot->memory_pressure != NULL;
> @@ -873,6 +900,17 @@ static inline bool sk_under_memory_pressure(const struct sock *sk)
> {
>   if (!sk->sk_prot->memory_pressure)
>   return false;
> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
> + if (static_branch(&memcg_socket_limit_enabled)) {
> + struct cg_proto *cg_proto = sk->sk_cgrp;
> +
> + if (!cg_proto)
> +   goto nocgroup;
> + return !!*cg_proto->memory_pressure;
> + } else

What is dangling 'else' for ?

```
> +nocgroup:
> +#endif
> +
>   return !!*sk->sk_prot->memory_pressure;
>  }
>
> @@ -880,52 +918,176 @@ static inline void sk_leave_memory_pressure(struct sock *sk)
>  {
>   int *memory_pressure = sk->sk_prot->memory_pressure;
>
> - if (memory_pressure && *memory_pressure)
> + if (!memory_pressure)
> +  return;
> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
> + if (static_branch(&memcg_socket_limit_enabled)) {
> +  struct cg_proto *cg_proto = sk->sk_cgrp;
> +
> +  if (!cg_proto)
> +   goto nocgroup;
> +
> +  for (; cg_proto; cg_proto = cg_proto->parent)
> +   if (*cg_proto->memory_pressure)
> +    *cg_proto->memory_pressure = 0;
> + }
> +nocgroup:
> +#endif
```

Hmm..can't we have a good way for avoiding this #ifdef ?

I guess... as NUMA_BUILD macro in page_alloc.c, you can define

if (HAS_KMEM_LIMIT && static_branch(&.....)).

For example,
==
```
#include <stdio.h>

#define HAS_SPECIAL     0

int main(int argc, char *argv[])
{
        if (HAS_SPECIAL)
                call();

        printf("Hey!");
}
```

==

This can be compiled.

So. I guess...

```
#ifdef CONFIG_CGROUP_MEM_RES_CTLR
#define do_memcg_kmem_account static_branch(&memcg_socket_limit_enabled)
#else
#define do_memcg_kmem_account 0
#endif
```

maybe good.(not tested.)

BTW, I don't think 'goto nocgroup' is good.

```
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index 3becb24..12a08bf 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -377,6 +377,40 @@ enum mem_type {
>  #define MEM_CGROUP_RECLAIM_SOFT_BIT 0x2
>  #define MEM_CGROUP_RECLAIM_SOFT  (1 << MEM_CGROUP_RECLAIM_SOFT_BIT)
>
> +static inline bool mem_cgroup_is_root(struct mem_cgroup *memcg)
> +{
> + return (memcg == root_mem_cgroup);
> +}
> +
```

Why do you need this move of definition ?

Thanks,
-Kame