
Subject: Re: [PATCH 13/23] slub: create duplicate cache
Posted by [Suleiman Souhlal](#) on Mon, 30 Apr 2012 20:15:23 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Sun, Apr 22, 2012 at 4:53 PM, Glauber Costa <glommer@parallels.com> wrote:

```
> This patch provides kmem_cache_dup(), that duplicates
> a cache for a memcg, preserving its creation properties.
> Object size, alignment and flags are all respected.
>
> When a duplicate cache is created, the parent cache cannot
> be destructed during the child lifetime. To assure this,
> its reference count is increased if the cache creation
> succeeds.
>
> Signed-off-by: Glauber Costa <glommer@parallels.com>
> CC: Christoph Lameter <cl@linux.com>
> CC: Pekka Enberg <penberg@cs.helsinki.fi>
> CC: Michal Hocko <mhocko@suse.cz>
> CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> CC: Johannes Weiner <hannes@cmpxchg.org>
> CC: Suleiman Souhlal <suleiman@google.com>
> ---
> include/linux/memcontrol.h | 3 +++
> include/linux/slab.h       | 3 +++
> mm/memcontrol.c           | 44
+++++++
> mm/slub.c                 | 37 +++++
> 4 files changed, 87 insertions(+), 0 deletions(-)
>
> diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
> index 99e14b9..493ecdd 100644
> --- a/include/linux/memcontrol.h
> +++ b/include/linux/memcontrol.h
> @@ -445,6 +445,9 @@ int memcg_css_id(struct mem_cgroup *memcg);
> void mem_cgroup_register_cache(struct mem_cgroup *memcg,
>                                struct kmem_cache *s);
> void mem_cgroup_release_cache(struct kmem_cache *cachep);
> +extern char *mem_cgroup_cache_name(struct mem_cgroup *memcg,
> +                                   struct kmem_cache *cachep);
> +
> #else
> static inline void mem_cgroup_register_cache(struct mem_cgroup *memcg,
>                                              struct kmem_cache *s)
> diff --git a/include/linux/slab.h b/include/linux/slab.h
> index c7a7e05..909b508 100644
> --- a/include/linux/slab.h
> +++ b/include/linux/slab.h
> @@ -323,6 +323,9 @@ extern void *__kmalloc_track_caller(size_t, gfp_t, unsigned long);
```

```

>
> #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
> #define MAX_KMEM_CACHE_TYPES 400
> +extern struct kmem_cache *kmem_cache_dup(struct mem_cgroup *memcg,
> +
> +                struct kmem_cache *cachep);
> +void kmem_cache_drop_ref(struct kmem_cache *cachep);
> #else
> #define MAX_KMEM_CACHE_TYPES 0
> #endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index 0015ed0..e881d83 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -467,6 +467,50 @@ struct cg_proto *tcp_proto_cgroup(struct mem_cgroup *memcg)
> EXPORT_SYMBOL(tcp_proto_cgroup);
> #endif /* CONFIG_INET */
>
> +/*
> + * This is to prevent races against the kmemalloc cache creations.
> + * Should never be used outside the core memcg code. Therefore,
> + * copy it here, instead of letting it in lib/
> + */
> +static char *kasprintf_no_account(gfp_t gfp, const char *fmt, ...)
> +{
> +    unsigned int len;
> +    char *p = NULL;
> +    va_list ap, aq;
> +
> +    va_start(ap, fmt);
> +    va_copy(aq, ap);
> +    len = vsnprintf(NULL, 0, fmt, aq);
> +    va_end(aq);
> +
> +    p = kmemalloc_no_account(len+1, gfp);
> +    if (!p)
> +        goto out;
> +
> +    vsnprintf(p, len+1, fmt, ap);
> +
> +out:
> +    va_end(ap);
> +    return p;
> +}
> +
> +char *mem_cgroup_cache_name(struct mem_cgroup *memcg, struct kmem_cache *cachep)
> +{
> +    char *name;
> +    struct dentry *dentry = memcg->css.cgroup->dentry;

```

Do we need `rcu_dereference()` here (and make sure we have `rcu_read_lock()`)?

This might need to be done in all the other places in the patchset that get the memcg's dentry.

-- Suleiman
