Subject: Re: [RFC] memory controller : backgorund reclaim and avoid excessive locking [5/5] lazy page_cgroup
Posted by yamamoto on Mon, 18 Feb 2008 01:58:40 GMT
View Forum Message <> Reply to Message

> + /*
> + * For lazy freeing (not GC)
> + */
> + struct {
> + struct mem_cgroup_per_zone *mz;
> + int   num;
> +#define GARBAGE_MAXSIZE (16)
> + struct page_cgroup *vec[GARBAGE_MAXSIZE];
> + } garbage[NR_CPUS];
> };

i think you want to dedicate cache lines.

> @@ -176,12 +185,14 @@ struct page_cgroup {
>   struct list_head lru; /* per cgroup LRU list */
>   struct page *page;
>   struct mem_cgroup *mem_cgroup;
> + struct mem_cgroup_per_zone *mz; /* now belongs to...*/

is this for performance?

> @@ -408,10 +427,12 @@ static void __mem_cgroup_move_lists(stru
>   if (active) {
>   MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_ACTIVE) += 1;
>   pc->flags |= PAGE_CGROUP_FLAG_ACTIVE;
> + pc->mz = mz;
>   list_move(&pc->lru, &mz->active_list);
>   } else {
>   MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_INACTIVE) += 1;
>   pc->flags &= ~PAGE_CGROUP_FLAG_ACTIVE;
> + pc->mz = mz;
>   list_move(&pc->lru, &mz->inactive_list);
>   }
> }

isn't pc->mz already assigned by __mem_cgroup_add_list?

> @@ -1050,11 +1114,15 @@ mem_cgroup_force_empty_list(struct mem_c
>   if (list_empty(list))
>   return;
> retry:
> + all_free_garbages(mem);
>   count = FORCE_UNCHARGE_BATCH;

>   spin_lock_irqsave(&mz->lru_lock, flags);
>
>   while (--count && !list_empty(list)) {
>   pc = list_entry(list->prev, struct page_cgroup, lru);
> + /* If there are still garbage, exit and retry */
> + if (pc->flags & PAGE_CGROUP_FLAG_GARBAGE)
> +   break;

i think mem_cgroup_isolate_pages needs a similar check.

YAMAMOTO Takashi

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers